BRX Do-more! Communications

Chapter 13

In This Chapter...

Communications

Overview

The purpose of this chapter is to help the user gain an understanding of the communications capabilities of the BRX MPU.

The BRX MPU is capable of both serial and Ethernet communications using a wide variety of protocols. Listed in the table below are the supported methods and protocols.

Communications	
Serial RS-232/RS-485	**Ethernet**
Do-more! Protocol (Server)	Do-more! Protocol (Client, Server)
Modbus RTU (Client, Server)	Modbus TCP (Client, Server)
K-Sequence (Server)	HOST ECOM Protocol (Client, Server)
ASCII (In & Out)	EtherNet/IP (Explicit Messaging (Client, Server))
	Ethernet Remote I/O
	SMTP (Email)
	SNTP (Time Server)
	TCP Raw Packet
	UDP Raw Packet

Terminology

During the course of this chapter we will use terminology and phrases that are specific to a Protocol or Physical Medium the user should understand. By way of explanation we have included some common terms and definitions in this section. Definitions and explanations of specific parameters particular to each Protocol will be discussed later in this chapter.

Physical Medium – Wires, radios, cellular service, or satellite link. The physical method (hardware) on which the data is being transmitted or received. The physical medium contains no data information. Examples of a physical medium are: RS-232, RS-485 and Ethernet 10/100 Base T.

Protocol – A Protocol is the specification for the formatting of the data (bits, bytes and words) being transmitted through the physical medium. Examples of some common industry protocols are: Modbus RTU, Modbus TCP, K-Sequence, DirectNet and EtherNet/IP.

One way to think of the Physical Medium and the Protocol is to liken them to placing a phone call. The phone call is being placed over wires, cellular service or even perhaps a satellite link. This is the physical medium. Now if the call was to China, you would say “Hello” in English. If the person on the other end understands English, they will respond with “Hello”. If the person on the other end only understands Mandarin Chinese they might respond “Ni Hao”. If you do not understand Chinese, you will be confused as to what they are saying. This is the same for a Protocol. If your PLC uses Modbus TCP/IP and you try to talk to a PLC that only understands EtherNet/IP, then you will not be able to communicate with it. The Protocols must match in order to communicate.

Client (Master) – A Client is a Master device that requests data from a Server (Slave) device.

Server (Slave) – A Server is a Slave device that responds to a request from a Client (Master) device.

UDP – User Datagram Protocol (UDP) is part of the Transport Layer of the Internet Protocol Suite. The Transport Layer is Layer 6. UDP is a simple connectionless transport mechanism for Ethernet packets. Checksums are used for data integrity, however it has no error correction or guarantee of delivery. It is considerably faster than TCP/IP due to these factors. It is widely used when data is time sensitive because dropped packets may be preferable to retries and delays.

TCP – TCP or TCP/IP is part of the Transport Layer of the Internet Protocol Suite. The Transport Layer is Layer 6. TCP is defined as a reliable, ordered and an error checked delivery method. TCP/IP is most often used when the data integrity is more important than raw speed.

Field Device – A device external to the BRX MPU.

General Concepts

The BRX Do-more! MPU is capable of both serial and Ethernet communications to a wide variety of field devices such as HMIs, SCADA systems, PLCs, barcode readers and scales. The following sections of this chapter will be of significant help to you when connecting and communicating with the various protocols needed for these types of devices. For advanced users it is possible to write custom protocols using the raw commands provided in the Do-more! Designer software. So a thorough working knowledge of the protocol is required to accomplish this.

The BRX Do-more! MPU has tightly coupled security features incorporated into it to protect your installation. See the manual section for each protocol and also the Do-more! Designer software help file for more information on password protection and Protocol Specific Memory.

Multiple user accounts are supported and logged when accessed. When programming, session based communication with unique IDs is utilized to prevent unauthorized access.

The Do-more! Protocol can access the full memory structure and is capable of utilizing a security account to protect the data. Some HMIs and SCADA software such as C-more, C-more Micro and Point of View, can utilize the Do-more! Protocol for enhanced security while accessing the full memory area.

Protocol Specific memory areas are blocked out of the User Memory when using Modbus RTU, Modbus TCP/IP, HOST ECOM and K-Sequence protocols. These memory areas are only accessible externally by using the specific protocol that corresponds to the type of memory being accessed. They are usable by the programmer to pass data externally when using third party devices. This is discussed in more detail in each Protocol section later in this chapter.

USB Communications

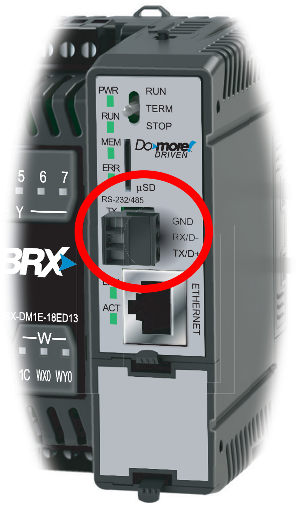
The POM slot USB port is useful for programming the BRX Do-more! MPU. It is compatible with Desktop PCs and Laptops utilizing USB 2.0 or higher, communicating through USB Type A to USB Type B cable. It does not support connections to other USB devices such as printers.

The POM slot USB port relies on drivers included with Windows, so there are no drivers to download. It truly is plug and play.

Serial Communications

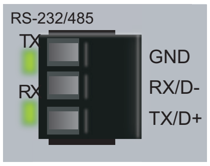
The RS-232/485 port is a removable three-pin screw terminal block located on the front of the CPU. This port is software selectable to communicate as RS-232 or as RS-485. In the RS-485 mode you can also enable a 120 Ohm termination resistor if needed.

The RS-232/485 port can be connected to the Do-more! Designer programming software, Modbus RTU master or slave devices, DirectLogic PLCs via K-Sequence protocol, as well as devices that send or receive non-sequenced ASCII strings or characters.



RS-232

RS-232 is a single point wiring standard that can be used to connect external devices to the PLC.



Pinout	RS232
1	GND
2	RXD
3	TXD

Built-in RS-232 Specifications	
Port Name	RS-232/RS-485
Description	Non-isolated Serial port that can communicate via RS-232 or RS-485 (software selectable). Includes ESD protection and built-in surge protection.
Supported Protocols	Do-more!™ Protocol (Default) Modbus RTU (Client & Server) K-Sequence (Slave) ASCII (In & Out)
Data Rates	1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200 Baud
Default Settings	RS-232, 115200bps, No Parity, 8 Data Bits, 1 Stop Bit, Station #1
Port Status LED	Green LED is illuminated when active for TXD and RXD
Port Type	Removable 3-pin terminal strip 3.5 mm pitch
RS-232 TXD	RS-232 Transmit output
RS-232 RXD	RS-232 Receive input
RS-232 GND	Logic ground
RS-232 Maximum Output Load (TXD/RTS)	3kΩ, 1000pf
RS-232 Minimum Output Voltage Swing	±5V
RS-232 Output Short Circuit Protection	±15mA
Cable Requirements	RS-232 use P/N L19772-XXX from automationdirect.com
Maximum Distance	30 meters (100 feet); 6 meters (20 foot) recommended maximum
Replacement Connector	ADC Part # BX-RTB03S

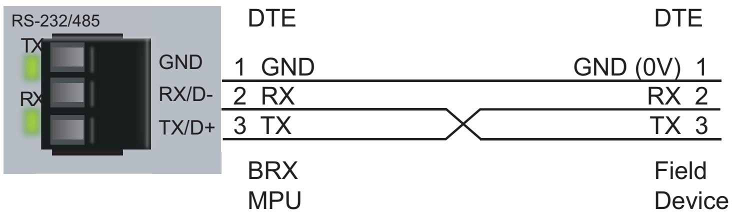
Serial Communications, continued

The manner in which external devices are wired to the CPU depend on whether the device is considered to be Data Terminal Equipment (DTE) or Data Communications Equipment (DCE). The CPU is considered a DTE device. Most Modbus or ASCII devices being connected to the CPU will also be considered a DTE device and will need to swap TX and RX (as shown below), but you should always consult the documentation of that device to verify.

If a device such as a Modem, which is a DCE device, is placed between the CPU and another Modbus or ASCII device it will most likely require connecting the signals straight across (TX to TX and RX to RX). Again, this can differ from manufacturer to manufacturer so always consult the documentation before wiring the devices together.

RS-232 is a ground referenced signal and as such it is susceptible to noise and ground differentials which may make it unsuitable for use in some circumstances.

The wiring for RS-232 is shown below. Please note that there are no connections for RTS (Ready To Send), CTS (Clear To Send) or for port powered devices (+5VDC).



NOTE: *Recommended distance between RS-232 devices is less than 6 meters (20 feet) in industrial environments. For longer distances, please consider using RS-485 or Ethernet.*

RS-485

RS-485 is a multi-point wiring standard that can be used to connect external devices to the PLC.

Built-in RS-485 Specifications

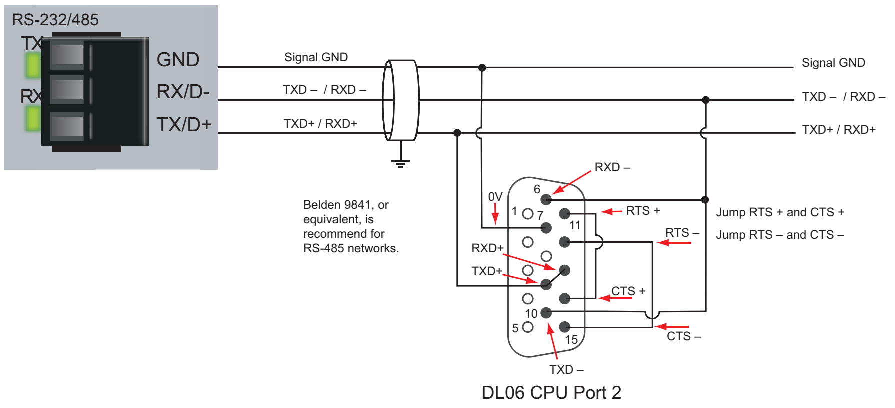
Port Name	RS-232/RS-485
Description	Non-isolated Serial port that can communicate via RS-232 or RS-485 (software selectable). Includes ESD protection and built-in surge protection.
Supported Protocols	Do-more!™ Protocol (Default) Modbus RTU (Client & Server) K-Sequence (Slave) ASCII (In & Out)
Data Rates	1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200 Baud
Default Settings	RS-232, 115200bps, No Parity, 8 Data Bits, 1 Stop Bit, Station #1
Port Status LED	Green LED is illuminated when active for TXD and RXD
Port Type	Removable 3-pin terminal strip 3.5 mm pitch
RS-485 Station Addresses	1–247
RS-485 TXD-/RXD-	RS-485 transceiver low
RS-485 TXD+/RXD+	RS-485 transceiver High
RS-485 GND	Logic ground
RS-485 Input Impedance	19kΩ
RS-485 Maximum Load	50 transceivers, 19kΩ each, 120Ω termination
RS-485 Output Short Circuit Protection	±250mA, thermal shut-down protection
RS-485 Electrostatic Discharge Protection	±8kV per IEC1000-4-2
RS-485 Electrical Fast Transient Protection	±2kV per IEC1000-4-4
RS-485 Minimum Differential Output Voltage	1.5 V with 60Ω load
RS-485 Fail Safe Inputs	Logic high input state if inputs are unconnected
RS-485 Maximum Common Mode Voltage	-7.5 V to 12.5 V
Cable Requirements	RS-485 use P/N L19827-XXX from automationdirect.com
Maximum Distance	1000 meters (3280 feet)
Replacement Connector	ADC Part # BX-RTB03S

The RS-485 port is useful for connecting multiple devices on one network and/or connecting devices to the CPU at much longer distances than RS-232. The RS-485 standard supports distances of up to 1000 meters without requiring a repeater. The distance can be increased by placing an RS-485 repeater on the network, if necessary. The RS-485 Port on the CPU can support up to 50 devices, depending on the load of each device (assuming a 19kΩ load for each device).

RS-485 utilizes a differential signal which makes it much more immune to noise and grounding issues than RS-232 and is therefore a much better choice when available for communications.

This port only supports RS-485, 2-wire connections. For 4-wire RS-485 or RS-422, a converter, such as an FA-ISOCON must be used.

The wiring for RS-485 is shown below. Please note that there are no connections for RTS (Ready To Send), CTS (Clear To Send) or for port powered devices (+5VDC).

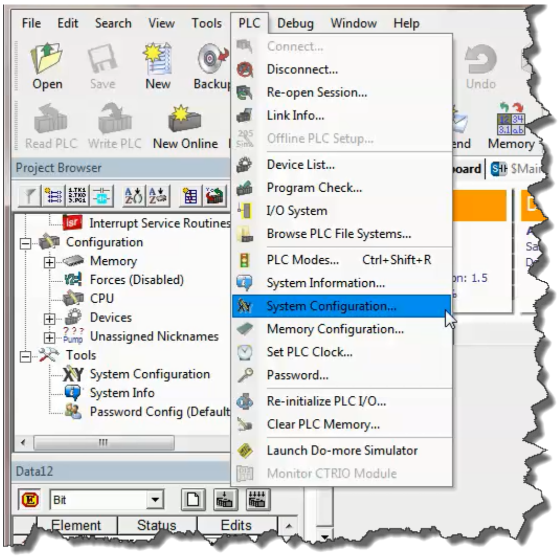


Serial Port Settings

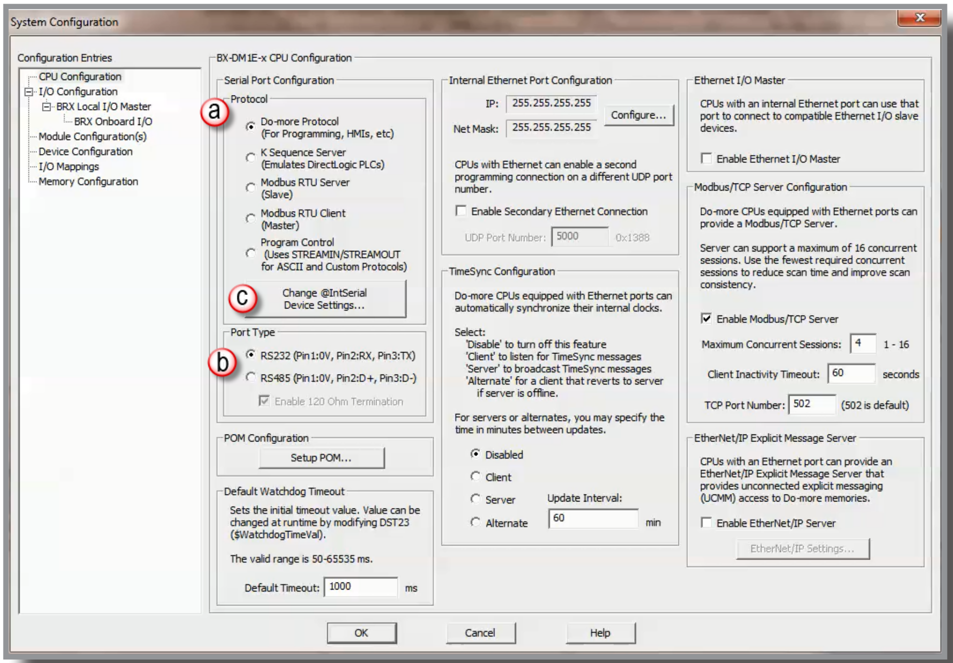
Setting up your serial port is straightforward. Follow the simple steps below.

1. You must know what physical medium your connecting device uses.
2. Establish which protocol your device supports.
3. Wire the physical medium using the wiring diagrams for either RS-232 or RS-485 on the previous pages.
4. Select supported protocol from the configuration list in the Do-more! Designer software.
5. Select baud rate, data bits, stop bits and parity for your device.
6. Set the baud, data bits, stop bits and parity to match your device and Do-more! Designer software.
7. If your device is a Client device such as an HMI, then your setup is completed.
8. If your device is a Server device and the BRX Do-more! MPU will be requesting data from it, then some ladder logic must be written. Please refer to the specific section for your chosen protocol. The Do-more! Designer software help file is also a good resource for setting up communications.

To set up your serial communications port in Do-more! Designer, from the PLC drop-down menu at the top of the screen, select **System Configuration**.



The *System Configuration* dialog will appear. Within this dialog box you can (**a**) pick the ***Protocol*** and (**b**) ***Port Type***, whether the port should be used for RS-232 or RS-485.



System Configuration

Configuration Entries
- CPU Configuration
- I/O Configuration
  - BRX Local I/O Master
    - BRX Onboard I/O
- Module Configuration(s)
- Device Configuration
- I/O Mappings
- Memory Configuration

BX-DM1E-x CPU Configuration

Serial Port Configuration

Protocol
- [✓ Do-more Protocol (For Programming, HMIs, etc)]
- K Sequence Server (Emulates DirectLogic PLCs)
- Modbus RTU Server (Slave)
- Modbus RTU Client (Master)
- Program Control (Uses STREAMIN/STREAMOUT for ASCII and Custom Protocols)

Change @IntSerial Device Settings...

Port Type
- [✓ RS232 (Pin1:0V, Pin2:RX, Pin3:TX)]
- RS485 (Pin1:0V, Pin2:D+, Pin3:D-)
- [✓ Enable 120 Ohm Termination]

POM Configuration

Setup POM...

Default Watchdog Timeout

Sets the initial timeout value. Value can be changed at runtime by modifying DST23 ($WatchdogTimeVal).

The valid range is 50-65535 ms.

Default Timeout: 1000 ms

Internal Ethernet Port Configuration

IP: 255.255.255.255

Net Mask: 255.255.255.255

Configure...

CPUs with Ethernet can enable a second programming connection on a different UDP port number.

Enable Secondary Ethernet Connection

UDP Port Number: 5000 0x1388

TimeSync Configuration

Do-more CPUs equipped with Ethernet ports can automatically synchronize their internal clocks.

Select:
'Disable' to turn off this feature
'Client' to listen for TimeSync messages
'Server' to broadcast TimeSync messages
'Alternate' for a client that reverts to server if server is offline.

For servers or alternates, you may specify the time in minutes between updates.

- [✓ Disabled]
- Client
- Server
- Alternate

Update Interval: 60 min

Ethernet I/O Master

CPUs with an internal Ethernet port can use that port to connect to compatible Ethernet I/O slave devices.

Enable Ethernet I/O Master

Modbus/TCP Server Configuration

Do-more CPUs equipped with Ethernet ports can provide a Modbus/TCP Server.

Server can support a maximum of 16 concurrent sessions. Use the fewest required concurrent sessions to reduce scan time and improve scan consistency.

[✓ Enable Modbus/TCP Server]

Maximum Concurrent Sessions: 4 1 - 16

Client Inactivity Timeout: 60 seconds

TCP Port Number: 502 (502 is default)

EtherNet/IP Explicit Message Server

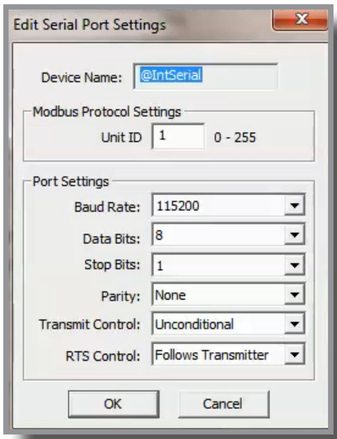
CPUs with an Ethernet port can provide an EtherNet/IP Explicit Message Server that provides unconnected explicit messaging (UCMM) access to Do-more memories.

Enable EtherNet/IP Server

EtherNet/IP Settings...

OK Cancel Help

Once you have selected the desired Protocol and Port Type, click (**c**) the button labeled **Change @IntSerial Device Settings**.



Edit Serial Port Settings

Device Name: @IntSerial

Modbus Protocol Settings

Unit ID 1 0 - 255

Port Settings

Baud Rate: 115200

Data Bits: 8

Stop Bits: 1

Parity: None

Transmit Control: Unconditional

RTS Control: Follows Transmitter

OK Cancel

The *Edit Serial Port Settings* dialog (right) will appear. Here you will configure the serial port settings to match any additional device(s) settings. These settings must match exactly in order for the devices to communicate properly.

Unit ID – This is the Modbus protocol identification number when the BRX Do-more! MPU functions as a server device. For RS-232 this value should always be set to 1. For RS-485, this value will depend on how many other devices are present on the network and how they are to be numbered.

Baud Rate – The rate at which the data is transmitted. The higher the number the faster the data will be transmitted. Choosing a lower value can help with issues when the data is not being received reliably.

Available Baud Rate choices: 115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200

Data Bits – The number of bits in each character.

Available choices: 7 or 8.

Edit Serial Port Settings Parameters, continued

Stop Bits – The number of bits sent to denote the end of each character.

Available choices: 1 or 2.

Parity – The method of error detection used during transmission.

Available choices: None, Odd, Even

Transmit Control – Designates when data will be transmitted.

Available choices include:

- Unconditional – Data will be transmitted as soon as it reaches the output buffer
- Wait for CTS – Data will be transmitted when the CTS line is asserted.
- Delayed 5ms, delayed 50ms, delayed 250ms, delayed 500ms – After data reaches the output buffer, the RTS line will be asserted, and the transmitting of the data will be delayed by the selected number of milliseconds.

RTS Control – This setting is unused in BRX Do-more! MPUs and should always be set to **Follows Transmitter**.

Timeout – how many milliseconds should the instruction wait for the remote Modbus RTU Server to respond, this can be any constant from 0 to 32767.

Retries – how many times should the instruction retry the communication with the remote Modbus RTU Server, this can be any constant from 0 to 255.

Inter-packet Delay – the amount of time (in microseconds) the Modbus/RTU Client will place between packets as they are sent, this can be any be any constant value between 0 and 65535.

Serial Protocols

The BRX Do-more! MPU has several Protocol choices for communicating to external devices. In this section we will go over the choices and describe each choice.

Serial RS-232/RS-485
Do-more! Protocol (Server)
Modbus RTU (Client, Server)
K-Sequence (Server)
ASCII (In & Out)

Do-more! Protocol

The Do-more! protocol is a proprietary protocol that is used exclusively by the Do-more! family of controllers. This is a very feature rich and secure protocol that is used to communicate between the Do-more! Designer software and Do-more! controllers. It can also be used to communicate between multiple Do-more! controllers or to other devices such as the C-more HMI, and some SCADA systems such as Point of View support the Do-more! Protocol.

Modbus RTU

Modbus RTU is a protocol overseen by Modbus.org. This is an open standard, meaning that anyone can utilize it freely.

Modbus RTU can be utilized as either a client or server configuration. It supports multiple server (slave) devices on RS-485 and a single server (slave) device on RS-232.

Modbus RTU Server (Slave)

As a Modbus RTU Server (Slave), the BRX MPU is functioning as a listening/replying device. When an external Client (Master) device requests data from the BRX MPU, the MPU will reply with the appropriate data.

All Modbus data is stored in four sets of registers in the BRX MPU. This memory area is blocked off specifically for Modbus communications. You must place data in these registers in order for a Modbus device to be able to access it.

The Modbus data area is loosely data typed and casting or other instructions such as PUBLISH and SUBSCRIBE can be utilized to convert data in this area to the correct data type required.

Modbus		
Register Type	**Register Name**	**Range**
Holding Coil	MC	00000–01023
Input Coil	MI	10000–11023
Holding Register	MHR	30000–31047
Input Register	MIR	40000–42047

Please reference the Help file on casting, PUBLISH and SUBSCRIBE.

NOTE: *Ranges can be expanded in the Memory Configuration section of the Do-more! Designer software as needed.*

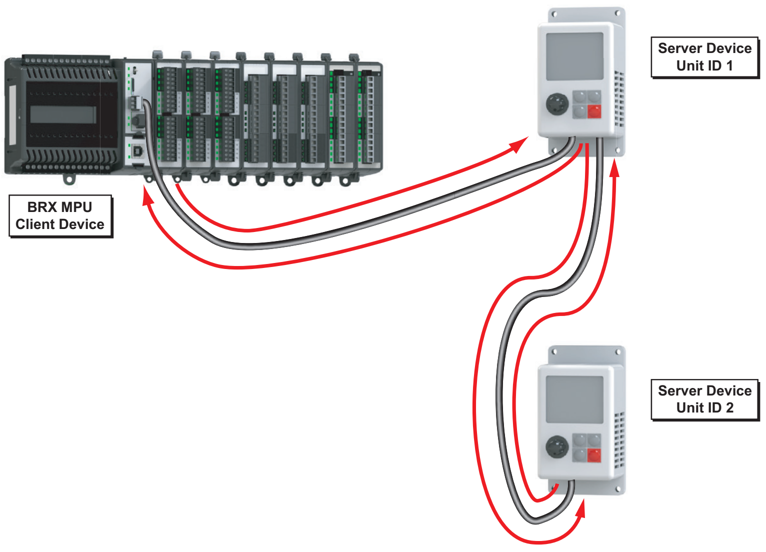
Serial Protocols, continued

The Modbus RTU Server (Slave) supports the following function codes:

Modbus	
Function Code	**Description**
1	Read coil
2	Read discrete inputs
3	Read holding registers
4	Read input registers
5	Write single coil
6	Write single registers
7	Read exception status
15	Write multiple coils
16	Write multiple registers
22	Mask write registers

Modbus RTU Client (Master)

As a Modbus RTU Client (Master), the BRX MPU is requesting data from a Modbus RTU Server (Slave) device. In order for this to work, you need to know quite a few things about your Server device such as the function codes that it supports, the data registers that are accessible and possibly the Unit ID or Slave Address.

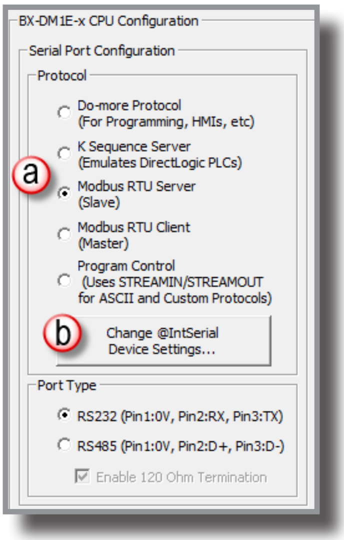


BRX MPU used as a Modbus RTU Client (Master) Example

Serial Protocols, continued

NOTE: *Only one Modbus RTU Client (Master) can be on a network.*

Opening the *System Configuration* dialog box you will find the **Serial Port Configuration** section where you can select the required protocol. Setting the serial port as (**a**) Modbus RTU Server (Slave), select (**b**) to create a the device @IntSerialDevice. This device will handle all of the communications with the external Modbus Servers (Slaves) through the serial port on the front of the BRX MPU.


BX-DM1E-x CPU Configuration

Serial Port Configuration

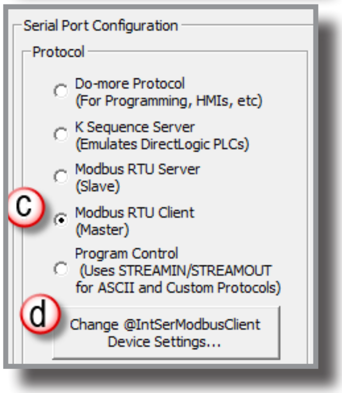
Protocol
- Do-more Protocol (For Programming, HMIs, etc)
- K Sequence Server (Emulates DirectLogic PLCs)
- [✓ Modbus RTU Server (Slave)]
- Modbus RTU Client (Master)
- Program Control (Uses STREAMIN/STREAMOUT for ASCII and Custom Protocols)

Port Type
- [✓ RS232 (Pin1:0V, Pin2:RX, Pin3:TX)]
- RS485 (Pin1:0V, Pin2:D+, Pin3:D-)

✓ Enable 120 Ohm Termination

As a Modbus RTU Client (Master), you do not have to sequence the MRX read and MWX write instructions. You can just drop them into your program and they will work in a round robin manner. However, sequencing the instructions will give you better control and allow you to build complex communication patterns to optimally communicate with your devices.

Selecting (c) **Modbus RTU Client** (**Master**) creates the device @IntSerModbusClient. Clicking on (**d**) brings up the *EditModbus RTU Client Settings* dialog box (below). Here you can configure Modbus Protocol Settings.

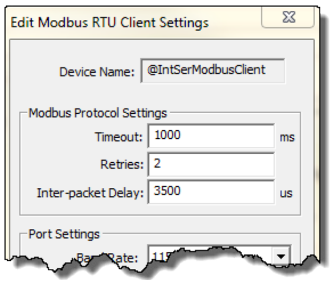

Serial Port Configuration

Protocol
- Do-more Protocol (For Programming, HMIs, etc)
- K Sequence Server (Emulates DirectLogic PLCs)
- Modbus RTU Server (Slave)
- [✓ Modbus RTU Client (Master)]
- Program Control (Uses STREAMIN/STREAMOUT for ASCII and Custom Protocols)

Timeout – Time in milliseconds that the instruction waits for the remote Modbus RTU Server to respond, this can be any constant from 0 to 32767.

Retries – The number of retries of the instruction to communicate with the remote Modbus RTU Server, this can be any constant from 0 to 255.

Inter-packet Delay – Time, in microseconds, that the Modbus/RTU Client will place between packets as they are sent, this can be any be any constant value between 0 and 65535.


Edit Modbus RTU Client Settings

Device Name: @IntSerModbusClient

Modbus Protocol Settings

Timeout: 1000 ms

Retries: 2

Inter-packet Delay: 3500 us

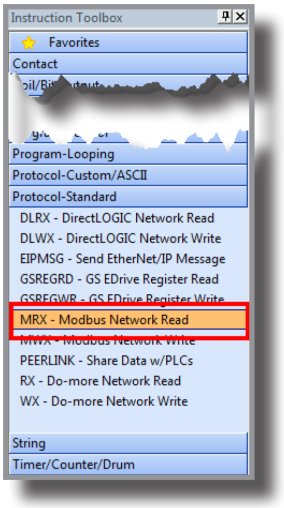
Port Settings

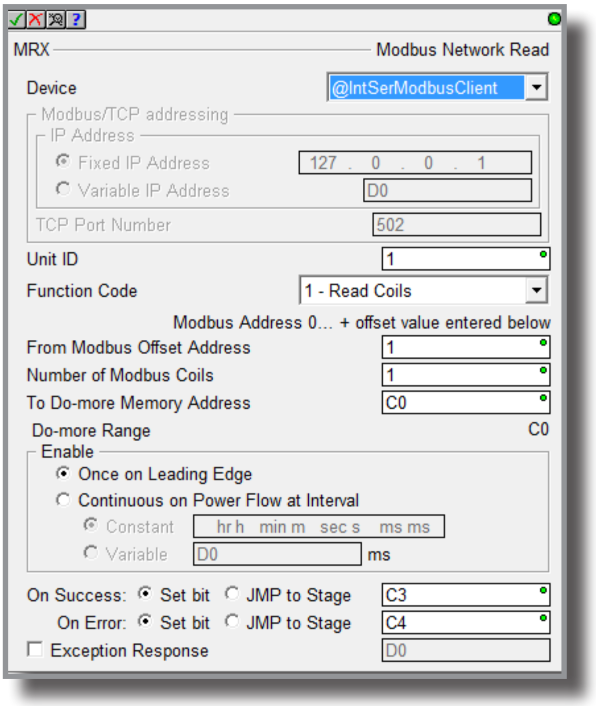
Baud Rate: 115

Communication instructions in Do-more! Designer have **Success** and **Error** feedback built in so that you can either set a bit or move to a Stage if you are doing stage style programming. There are examples of using Stage (state) programming in the software Help file that show how you could use this to implement a complex communications routine.

MRX Instruction

The MRX instruction, found in the *Instruction Tool Box* under the *Protocol-Standard* tab (right and below), is used to read from a Modbus RTU Server (Slave). (For more detailed information please refer to the Do-more! Designer Help files.)


Instruction Toolbox
Favorites
Contact
Program-Looping
Protocol-Custom/ASCII
Protocol-Standard
DLRX - DirectLOGIC Network Read
DLWX - DirectLOGIC Network Write
EIPMSG - Send EtherNet/IP Message
GSREGRD - GS EDrive Register Read
GSREGWR - GS EDrive Register Write
MRX - Modbus Network Read
MWX - Modbus Network Write
PEERLINK - Share Data w/PLCs
RX - Do-more Network Read
WX - Do-more Network Write
String
Timer/Counter/Drum


MRX — Modbus Network Read

Device: @IntSerModbusClient

Modbus/TCP addressing
IP Address
Fixed IP Address: 127 . 0 . 0 . 1
Variable IP Address: D0
TCP Port Number: 502

Unit ID: 1

Function Code: 1 - Read Coils

Modbus Address 0… + offset value entered below
From Modbus Offset Address: 1
Number of Modbus Coils: 1
To Do-more Memory Address: C0
Do-more Range: C0

Enable
[✓ Once on Leading Edge]
Continuous on Power Flow at Interval
[✓ Constant] hr h min m sec s ms ms
Variable D0 ms

On Success: [✓ Set bit] JMP to Stage C3
On Error: [✓ Set bit] JMP to Stage C4
Exception Response D0

Device – The device associated with the physical port that you want to communicate from. *@IntSerModbusClient* is the name of the device associated with the built in serial port when set as a Modbus Client (Master).

Unit ID – The ID number of the Server (Slave) device that the BRX MPU will talk to. Typically this is 1 for RS-232. For RS-485, this number could change depending on which device number you are talking to on the network.

Function Code – Select from the drop-down list one of the following Modbus function codes to use:

1 - Read Coils

2 - Read Discrete Inputs

3 - Read Holding Registers

4 - Read Input Registers

7 - Read Exception Status

MRX Instruction, continued

Modbus Address 0... + offset value entered below:

From Modbus Offset Address – The address in the Modbus Server (Slave) that you will be reading from. This address may be offset by a value of +1 depending on how the manufacturer followed the Modbus standard..

Number of Modbus Coils/ Registers – Based on the Function Code selected, this selection specifies how many consecutive elements to read. For example, if using function code 3, read holding registers, with an offset of 1, your request starts with MHR1, which corresponds to Modbus address 400001.

To Do-more Memory Address – Specifies the beginning address of a range of bits or numeric locations in the MPU where the data that is read will be stored. This data type (bit or register) must match the type expected by the Function Code.

Do-more Range – This is the ending register where the data will be stored at, calculated by taking the To Do-more Memory address and adding the Number of Modbus Coils/Registers value to it.

Enable – Designates how this instruction will operate. Select from one of the following:

Once on Leading Edge – Select this option to have this instruction run to completion exactly one time. Typically, this instruction will take more than one controller scan to complete. Configured this way the instruction is Edge Triggered.

Continuous on Power Flow at Interval – Select this option to have this instruction run as long as the instruction has power flow. After the instruction has initially run, if the instruction still has power flow, the instruction will remain enabled and will wait the specified amount of time before running again. The following options select how much time (in milliseconds) to wait between successive runs. A value of zero milliseconds (0ms) means the instruction will re-run immediately.

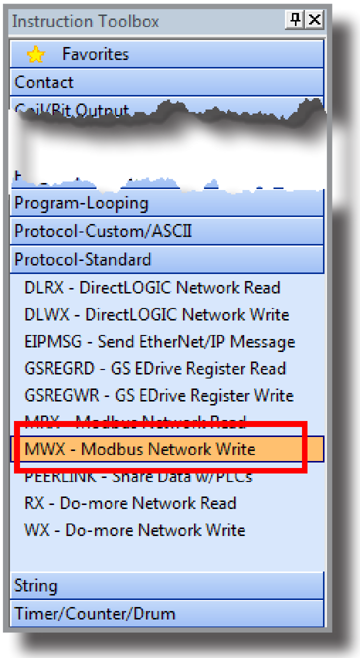
On Success – When the instruction completes successfully this action will be performed.

On Error – When the instruction does not successfully complete, this action will be performed.

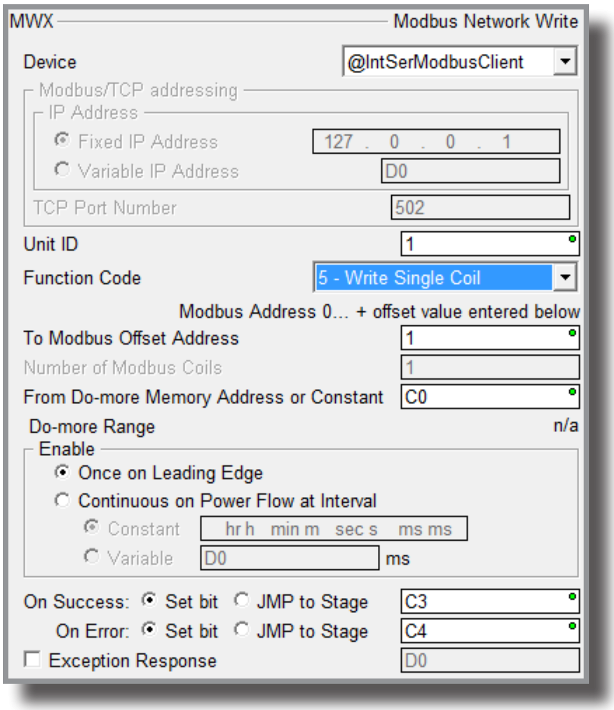
Exception Response – For errors where the message was received properly by the Server (Slave) device, this will contain a value to indicate why the Server rejected the message. This can aid in troubleshooting the issue.

MWX Instruction

The MWX instruction, found in the *Instruction Tool Box* under *Protocol - Standard* (right and below), is used to write to a Modbus RTU Server (Slave). (For specific information please refer to the Do-more! Designer help files.)



Instruction Toolbox
Favorites
Contact
Coil/Bit Output
Program-Looping
Protocol-Custom/ASCII
Protocol-Standard
DLRX - DirectLOGIC Network Read
DLWX - DirectLOGIC Network Write
EIPMSG - Send EtherNet/IP Message
GSREGRD - GS EDrive Register Read
GSREGWR - GS EDrive Register Write
MRX - Modbus Network Read
MWX - Modbus Network Write
PEERLINK - Share Data w/PLCs
RX - Do-more Network Read
WX - Do-more Network Write
String
Timer/Counter/Drum



MWX — Modbus Network Write

Device: @IntSerModbusClient

Modbus/TCP addressing
IP Address
(●) Fixed IP Address 127 . 0 . 0 . 1
( ) Variable IP Address D0
TCP Port Number 502

Unit ID: 1

Function Code: 5 - Write Single Coil

Modbus Address 0... + offset value entered below
To Modbus Offset Address: 1
Number of Modbus Coils: 1
From Do-more Memory Address or Constant: C0
Do-more Range: n/a

Enable
(●) Once on Leading Edge
( ) Continuous on Power Flow at Interval
(●) Constant: hr h min m sec s ms ms
( ) Variable: D0 ms

On Success: (●) Set bit ( ) JMP to Stage C3
On Error: (●) Set bit ( ) JMP to Stage C4
☐ Exception Response D0

Device – The device associated with the physical port that you want to communicate from. *@IntSerModbusClient* is the name of the device associated with the built in serial port when set as a Modbus Client (Master).

Unit ID – The ID number of the Server (Slave) device the BRX MPU is talking to. Typically this is 1 unless there are multiple devices on the network..

Function Code – Select from the drop-down list one of the following Modbus function codes to use:

5 - Write Single Coil

6 - Write Single Register

15 - Write Multiple Coils

16 - Write Multiple Registers

Modbus Address 0... + offset value entered below:

To Modbus Offset Address – The starting register to which you will be writing data. This address may be offset by a value of +1 depending on how the manufacturer followed the Modbus standard.

Number of Modbus Coils/ Registers – This selection specifies how many consecutive elements to write from the Modbus Offset Address. For example, if using function code 6,write a single register, with an offset of 1, it would write a value to Modbus address 400001.

From Do-more Memory Address – specifies the beginning address of a range of bits or numeric locations in the MPU to where the data will be written. This data type (bit or register) must match the type expected by the Function Code.

Do-more Range – This is the ending register from where the data will be written, calculated by taking the From Do-more! Memory address and adding the Number of Modbus Coils/ Registers value to it.

Enable – Designates how this instruction will operate. Select from one of the following:

Once on Leading Edge - Select this option to have this instruction run to completion exactly one time. Typically, this instruction will take more than one controller scan to complete. Configured this way the instruction is Edge Triggered.

Continuous on Power Flow at Interval - Select this option to have this instruction run as long as the instruction has power flow. After the instruction has initially run, if the instruction still has power flow, the instruction will remain enabled and will wait the specified amount of time before running again. The following options select how much time (in milliseconds) to wait between successive runs. A value of 0ms means the instruction will re-run immediately.

On Success – When the instruction completes successfully this action will be performed.

On Error – When the instruction does not complete successfully this action will be performed

Exception Response – For errors where the message was received properly by the Server (Slave) device, this will contain a value to indicate why the Server rejected the message. This can aid in troubleshooting the error issue.

K-Sequence

Server

The BRX Do-more! MPU can serve as a K-Sequence server to communicate to legacy devices that utilize the K-Sequence protocol such as DirectLogic PLC's, C-more HMI, SCADA systems, etc.

All K-Sequence data is stored in four sets of registers in the BRX Do-more! MPU. The memory area is blocked off specifically for K-Sequence communications. You must place data in these registers in order for a K-Sequence Client device to be able to access it.

The K-Sequence data area is loosely data typed. Instructions such as Publish and Subscribe are used to communicate with a K-Sequence Client and then casting can be utilized to convert data in this area to the proper data type needed as well. Please see the Help file for more information on Casting, PUBLISH and SUBSCRIBE.

K-Sequence

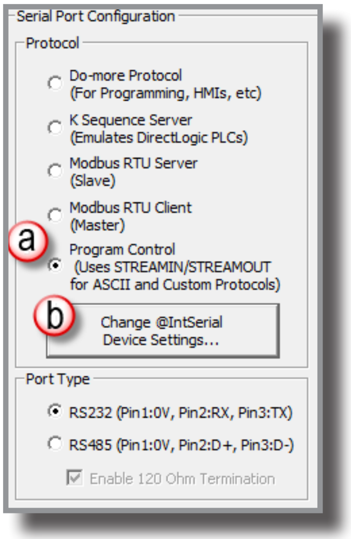
Register Type	Register Name	Range
Input Register	DLX	0–777
Output Register	DLY	0–777
Internal Coil Register	DLC	0–777
Internal Word Register	DLV	0–3777

NOTE: *Ranges can be expanded in the Memory Configuration section of the Do-more! Designer software as needed.*

ASCII

The BRX Do-more! MPU can communicate with devices that utilize a non-sequenced or a custom protocol. When using ASCII messaging it is important to know how the external device communicates. This will require knowledge of how the external device expects to send and receive the data. Most ASCII devices have specification guidelines in the user manual that explain the methods needed to facilitate communications. Having this reference handy can be invaluable, saving time while programming instead of struggling with getting the communications working.

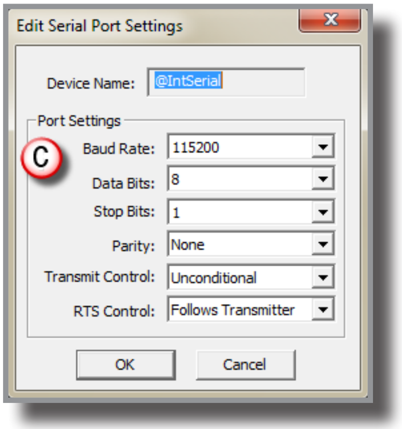
Utilizing ASCII as a messaging medium requires that the serial port (a) be set to **Program Control**. To configure the port settings select (b) the **Change @IntSerial Device Settings** button. This will bring up the (c) Edit Serial Port Settings dialog. **The communications settings here need to match those of the device you are communicating with.**


Serial Port Configuration

Protocol
- ( ) Do-more Protocol (For Programming, HMIs, etc)
- ( ) K Sequence Server (Emulates DirectLogic PLCs)
- ( ) Modbus RTU Server (Slave)
- ( ) Modbus RTU Client (Master)
- a (●) Program Control (Uses STREAMIN/STREAMOUT for ASCII and Custom Protocols)

b [Change @IntSerial Device Settings...]

Port Type
- (●) RS232 (Pin1:0V, Pin2:RX, Pin3:TX)
- ( ) RS485 (Pin1:0V, Pin2:D+, Pin3:D-)
- ☑ Enable 120 Ohm Termination

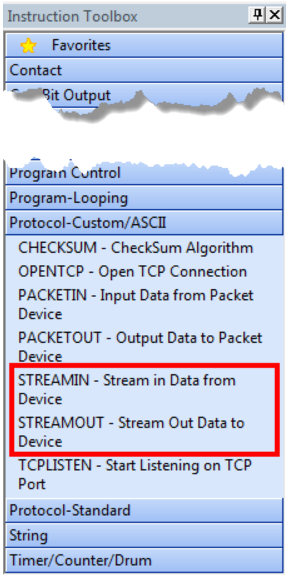

Edit Serial Port Settings

Device Name: @IntSerial

Port Settings (c)

Baud Rate:	115200
Data Bits:	8
Stop Bits:	1
Parity:	None
Transmit Control:	Unconditional
RTS Control:	Follows Transmitter

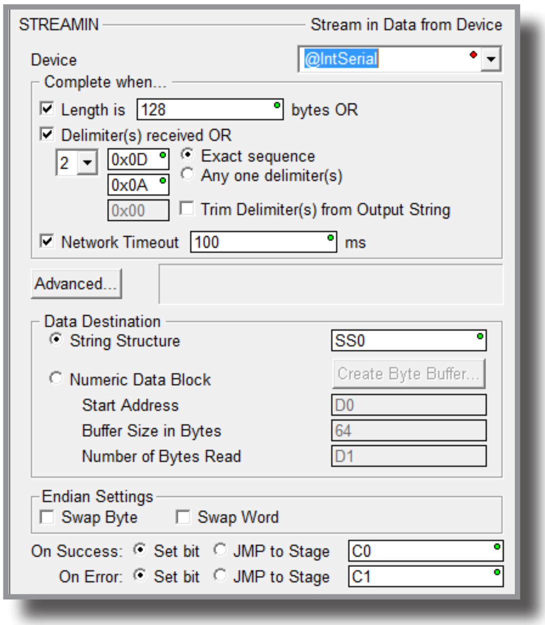
[OK] [Cancel]


Instruction Toolbox
- Favorites
- Contact
- Bit Output
- Program Control
- Program-Looping
- Protocol-Custom/ASCII
  - CHECKSUM - CheckSum Algorithm
  - OPENTCP - Open TCP Connection
  - PACKETIN - Input Data from Packet Device
  - PACKETOUT - Output Data to Packet Device
  - STREAMIN - Stream in Data from Device
  - STREAMOUT - Stream Out Data to Device
  - TCPLISTEN - Start Listening on TCP Port
- Protocol-Standard
- String
- Timer/Counter/Drum

Once set, the STREAMIN and STREAMOUT instructions (right) can be utilized.

One beneficial feature of the BRX Do-more! MPU is that the serial ports are buffered so that bi-directional data transfer is possible without needing the external device to pause between sending and receiving.

STREAMIN Instruction


STREAMIN — Stream in Data from Device

Device: @IntSerial

Complete when...
- [✓] Length is 128 bytes OR
- [✓] Delimiter(s) received OR
  - 2 — 0x0D, 0x0A, 0x00
  - (●) Exact sequence
  - ( ) Any one delimiter(s)
  - [ ] Trim Delimiter(s) from Output String
- [✓] Network Timeout 100 ms

Advanced...

Data Destination
- (●) String Structure: SS0
- ( ) Numeric Data Block — Create Byte Buffer...
  - Start Address: D0
  - Buffer Size in Bytes: 64
  - Number of Bytes Read: D1

Endian Settings
- [ ] Swap Byte
- [ ] Swap Word

On Success: (●) Set bit ( ) JMP to Stage — C0
On Error: (●) Set bit ( ) JMP to Stage — C1

The STREAMIN instruction is found in the Instruction Tool Box under the Protocol-Custom/ASCII tab.

Device – The device name associated with the physical port to which you want to communicate. @IntSerial is the name of the device associated with the built in serial port for ASCII control.

Complete when...

Length is...bytes OR – Specifies the number of characters received that will signal the completion of the instruction. The OR is used if **Delimiter(s) received OR** is selected.

Delimiter(s) received OR – Message characters that once received will signal the completion of the instruction. The OR implements:

Exact sequence – The specified characters must be receive in the order specified

Any one delimiter(s) – Receipt of any of the specified characters will signal completion.

Trim Delimiter(s) from Output String – Removes the delimiter characters from the Data Destination.

Network Timeout – The maximum amount of time that the instruction waits for completion.

Advanced – When selected a text box opens on the right. Allows the backspace character to remove characters from the received string before being placed into the Data Destination.

Data Destination

String Structure – The String memory location for the incoming data to be stored.

Numeric Data Block

Start Address – The offset into the Byte Buffer Data Block to store the incoming data.

Create Byte Buffer – Creates a data block of bytes to store the incoming data.

Buffer Size in Bytes – The maximum number of bytes that will be placed in the Byte Buffer Data Block.

Number of Bytes Read – Stores the number of bytes that were read into the Byte Buffer Data Block.

Endian Settings

Swap Byte – Swaps the bytes in each word of incoming data.

STREAMIN Instruction, Continued

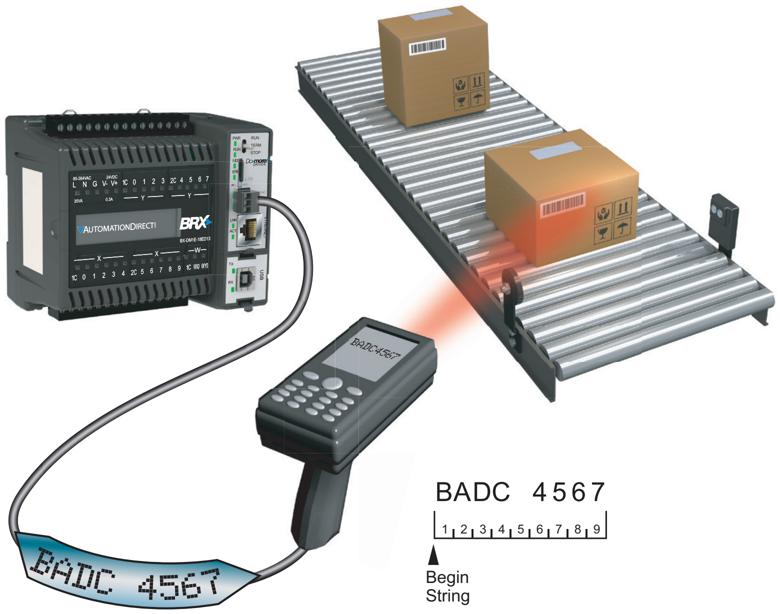
Swap Word – Swaps the words in each Double Word of incoming data.

On Success – When the instruction completes successfully this action will be performed.

On Error – When the instruction does not complete successfully this action will be performed.

Example using STREAMIN instruction.

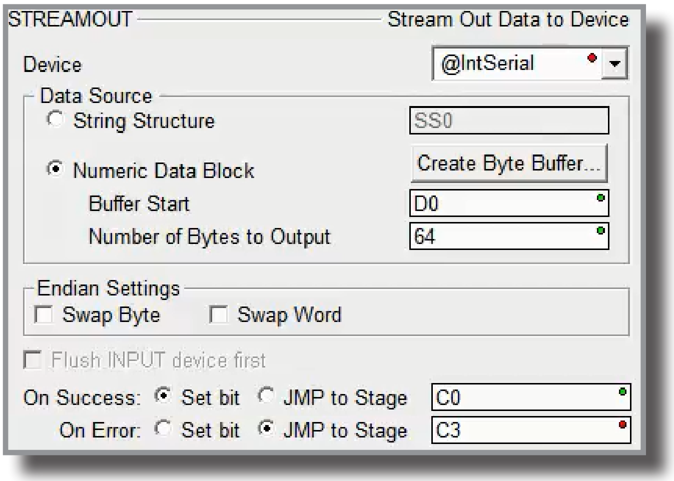
A barcode reader is an example of using STREAMIN and ASCII messaging. The string "BADC 4567" represented by the barcode is transmitted to the MPU as ASCII text with the use of the STREAMIN instruction.



__NOTE:__ Using @IntSerial.InQueue > 0 is the best way to trigger the STREAMIN function.

STREAMOUT Instruction

The STREAMOUT instruction is found in the Instruction Tool Box under the Protocol-Custom/ASCII tab.



STREAMOUT — Stream Out Data to Device

Device: @IntSerial

Data Source
- String Structure: SS0
- [✓ Numeric Data Block] — Create Byte Buffer...
  - Buffer Start: D0
  - Number of Bytes to Output: 64

Endian Settings
- ☐ Swap Byte ☐ Swap Word

☐ Flush INPUT device first

On Success: [✓ Set bit] JMP to Stage — C0

On Error: Set bit [✓ JMP to Stage] — C3

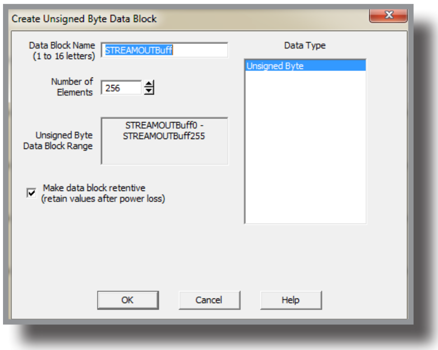
Device – The device associated with the physical port from which you want to communicate. ***@IntSerial*** is the name of the device associated with the built in serial port for ASCII control.

Data Source

String Structure – The String memory location that contains the data to be transmitted.

Numeric Data Block

- **Create Byte Buffer...** – Only available when **Numeric Data Block** is selected. Clicking this button opens ***Create Unsigned Byte Data Block*** box (right). Here you will name and create a data block of bytes to store the data to be transmitted.
- **Buffer Start** – The offset into the Byte Buffer Data Block for the data to be transmitted.
- **Number of Bytes to Output** – The number of bytes that will be transmitted.



Create Unsigned Byte Data Block

Data Block Name (1 to 16 letters): STREAMOUTBuff

Data Type: Unsigned Byte

Number of Elements: 256

Unsigned Byte Data Block Range: STREAMOUTBuff0 - STREAMOUTBuff255

☑ Make data block retentive (retain values after power loss)

OK Cancel Help

Endian Settings

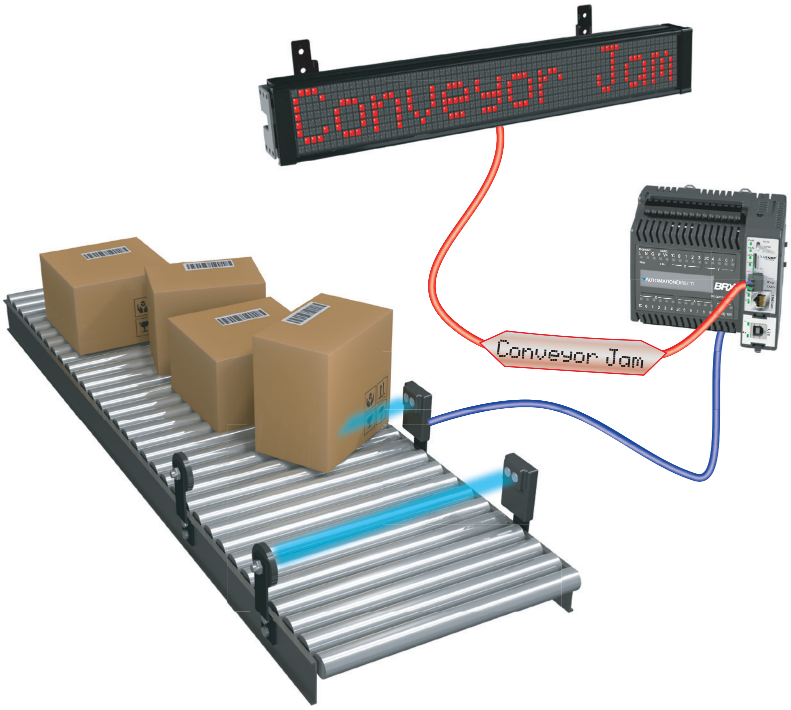
- **Swap Byte** – Swaps the bytes in each word of incoming data.
- **Swap Word** – Swaps the words in each Double Word of incoming data.

Flush INPUT device first – Clears the input buffer to ready it for a return response.

Example using STREAMOUT Instruction.

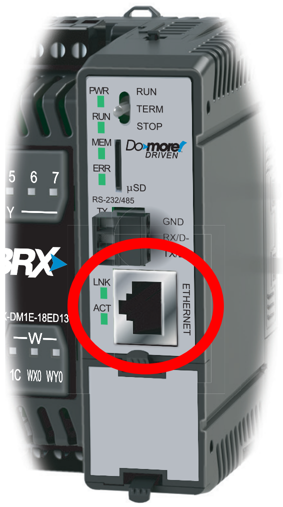
The example below uses the STREAMOUT instruction and ASCII messaging.

A sensor on the conveyor triggers a conveyor jam event in the program. The string "Conveyor Jam" is transmitted as ASCII text with the use of the STREAMOUT instruction.



Ethernet

The RJ-45 Ethernet port connector is located on the CPU faceplate. Rated at 10/100 Mbps, it accepts standard CAT5e cable and has built-in auto-crossover capability; no crossover cable is required.

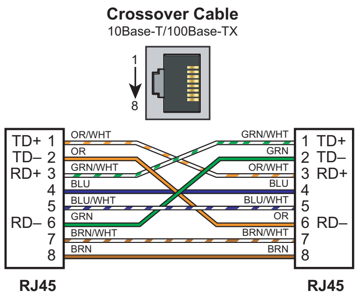


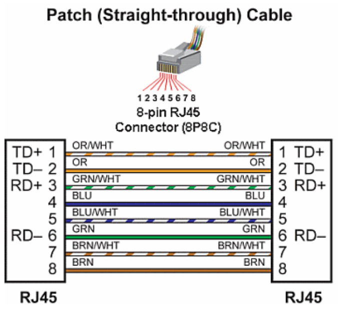
Ethernet Port Specifications

Port Name	ETHERNET
Ethernet Port Type	RJ45, CAT5e, 10/100 BASE-T, Auto Crossover
Description	Standard transformer isolated Ethernet port with built-in surge protection
Transfer Rate	10 Mbps (Orange LED) and 100 Mbps (Green LED)
Port Status LED	LED is solid when network LINK is established. LED flashes when port is active (ACT).
Supported Protocols	Do-more! Protocol Ethernet Remote I/O Modbus TCP/IP (Client & Server) EtherNet/IP (Explicit Messaging) HOST ECOM (DirectLogic) SMTP (Email), SNTP (Time Server) TCP/IP, UDP/IP (Raw packet)
Cable Recommendation	C5E-STxxx-xx from AutomationDirect.com
Ethernet Port Numbers	
Modbus TCP/IP	502 (configurable), TCP
EtherNet I/P (Explicit Messaging)	44818 (configurable), UDP
HOST ECOM	28784, UDP
Do-more! Protocol	28784, UDP

Wiring

The Ethernet port on the BRX Do-more! CPU's utilizes standard networking cables and devices. Category 5e cabling is preferred. Cables can be either patch (straight through) or crossover as the BRX Do-more! CPU Ethernet port has Auto MDI detection.





IP Addressing and Subnets

IP Addresses (used in conjunction with the Subnet Mask and Default Gateway address) are used for network routing. This allows for easy and logical separation of networks. It is outside of the scope of this user manual to explain how IP Addresses and Subnet masks are configured for actual usage. There are many books, documents and tools (Subnet calculators) on the Internet that provide this information. Each facility and network will incorporate their own rules and guidelines for how their networks are to be configured.

We suggest that users maintain their IP addressing and Subnets according to common networking practices by using private network addressing when at all possible. Examples of private addressing IP ranges can be found in the table below.

IP Addressing and Subnets

IP Address Range	Typical Subnet	Classful Description
192.168.0.0–192.168.255.255	255.255.0.0	Class C Network
172.16.0.0–172.31.255.255	255.240.0.0	Class B Network
10.0.0.0–10.255.255.255	255.0.0.0	Class A Network

***NOTE:** Notice that the IP range of 169.254.0.1 – 169.254.255.254 is not listed above. This range is reserved for APIPA addressing and is not recommended for use with most networks.*

Port Numbers

When doing TCP and UDP/IP communications, there is a Source Port number and Destination Port number for every message. The Client device must be aware of the Destination Port Number(s) the Server device is expecting to see, while the Server device must listen for this Destination Port number. After the Server device has received the message with the Destination Port Number on which it is listening, it will formulate the return message (if the applications require this) with the Source Port Number from the message sent as its Destination Port Number.

It is important to understand a little about the Port numbering concept because many Ethernet devices, such as routers with firewalls, will block messages with Destination Port numbers that are not configured for that device. Listed below are the default Port Numbers used in the BRX Do-more! platform. Some of the Port Numbers are configurable, allowing more flexibility when going through many different router applications.

Default Port Numbers

Ethernet	Port Numbers	TCP or UDP	Configurable
Peerlink	28784	UDP	No
Do-more! Protocol (Client, Server)	28784	UDP	Yes[1]
Modbus TCP (Client, Server)	502	TCP	Yes
HOST ECOM Protocol (Client, Server)	28784	UDP	No
EtherNet/IP (Explicit Messaging (Client, Server))	44818	TCP	Yes
Ethernet Remote I/O	28784	UDP	No
SMTP (Email)	25	TCP	Yes
SNTP (Time Server)	123	TCP	No
TCP Raw Packet	--	TCP	Yes
UDP Raw Packet	--	UDP	Yes

[1] Secondary Ethernet Connection can be enabled for the built-in Ethernet port of a BRX MPU. By default it has a UDP Port Number of 5000. It can be configured to any decimal number between 5000 and 65535, except for 28784.

While all of these protocols can be enabled on the BRX Do-more! CPU Ethernet port, it should be noted that Ethernet has a finite amount of traffic that can be handled. The amount of traffic load on the Ethernet port should be considered and polling times for Clients and Servers adjusted accordingly to allow ample time for the BRX Do-more! CPU to respond to and create requests for other devices.

NOTE: *Care should be taken when adding a BRX Do-more! MPU to an existing network as some protocols can create a significant traffic load.*

Ethernet Protocols

The BRX Do-more! MPU has several Ethernet Protocol choices for communicating to external devices. In this section we will go over the choices and describe each choice.

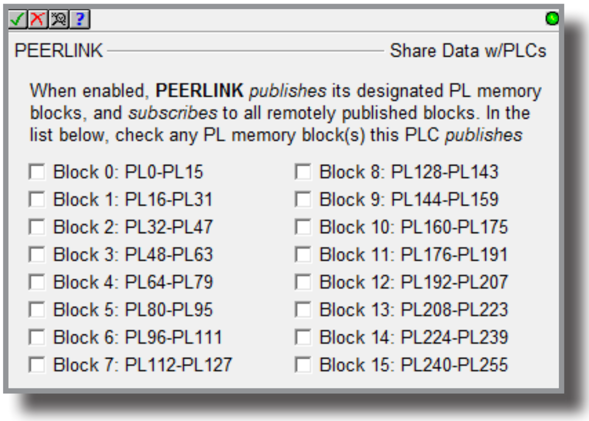
Ethernet Protocols
Peerlink
Do-more! Protocol (Client, Server)
Modbus TCP (Client, Server)
HOST ECOM Protocol (Client, Server)
EtherNet/IP (Explicit Messaging (Client, Server))
SMTP (Email)
SNTP (Time Server)
TCP Raw Packet
UDP Raw Packet

PEERLINK Instruction

The PEERLINK instruction allows easy data sharing across any PLC that is running Do-more! technology.

To set this instruction up, place your data into the appropriate PL (PEERLINK) registers that you wish this PLC to share to other Do-more! PLCs. Then check the box that corresponds to that memory area.

When this instruction is enabled, it will then automatically broadcast this information to all other Do-more! PLC's that are listening with a PEERLINK instruction.



PEERLINK — Share Data w/PLCs

When enabled, **PEERLINK** *publishes* its designated PL memory blocks, and *subscribes* to all remotely published blocks. In the list below, check any PL memory block(s) this PLC *publishes*

☐ Block 0: PL0-PL15	☐ Block 8: PL128-PL143
☐ Block 1: PL16-PL31	☐ Block 9: PL144-PL159
☐ Block 2: PL32-PL47	☐ Block 10: PL160-PL175
☐ Block 3: PL48-PL63	☐ Block 11: PL176-PL191
☐ Block 4: PL64-PL79	☐ Block 12: PL192-PL207
☐ Block 5: PL80-PL95	☐ Block 13: PL208-PL223
☐ Block 6: PL96-PL111	☐ Block 14: PL224-PL239
☐ Block 7: PL112-PL127	☐ Block 15: PL240-PL255

To listen for broadcasts from other Do-more! PLC's, all you have to do is place the instruction in your ladder code. If this PLC is not sharing data, do not check any boxes. Incoming data will automatically be placed into the same register area that the broadcasting PLC has checked.

Using PUBLISH and SUBSCRIBE instructions can help to get data into and out of the PL memory area as the PL memory area is untyped data. See the Do-more! Designer help file for more information on PEERLINK, PUBLISH and SUBSCRIBE instructions.

NOTE: *Keep in mind that multiple Do-more! PLC's cannot broadcast to the same memory area. This will cause an error.*

Do-more! Protocol

The Do-more! protocol is a proprietary protocol that is used exclusively by the Do-more! family of controllers. This is a very feature rich and secure protocol for communication with the Do-more! Designer software and between Do-more! controllers. It can also be used to communicate between multiple Do-more! controllers or between some brands of HMI's, such as C-more, to a Do-more! controller. Some SCADA systems such as POV also support the Do-more! Protocol.

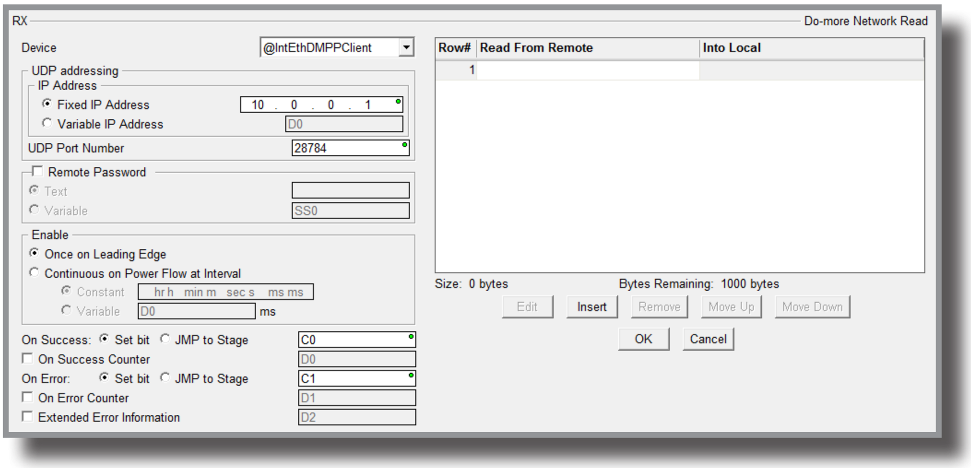
RX

The Do-more! Network Read (RX) instruction uses the Do-more! proprietary protocol to read incoming data on the on-board Ethernet port from another Ethernet-equipped Do-more! CPU. RX uses UDP (not TCP) protocol to communicate with the remote Do-more! CPU.

Each RX instruction can contain up to 50 individual read requests for a total of up to 1000 bytes of data. The RX instruction can read from all of the built-in memory blocks, all of the built-in structures, and any user-created memory blocks from the remote PLC. RX does NOT support reading a Heap Item from a remote Do-more! CPU.

In contrast to DirectLOGIC (DLRX / DLWX) and Modbus/TCP (MRX / MWX) network communication which only has access to the protocol-specific memory blocks in the remote CPUs, Do-more! RX has direct access to nearly all of the memory in the remote Do-more! CPU - including direct access to the CPU I/O memory.

The RX instruction establishes a session with the remote Do-more! CPU. The session is established using one of the User Accounts on the remote Do-more! CPU. It is through that User Account (System Security) that any access restrictions on what can be accessed can be enforced. By default it will use the Default User account (no password required).


RX — Do-more Network Read

Device: @IntEthDMPPClient

UDP addressing
IP Address
[✓ Fixed IP Address] 10 . 0 . 0 . 1
Variable IP Address D0
UDP Port Number 28784

Remote Password
[✓ Text]
Variable SS0

Enable
[✓ Once on Leading Edge]
Continuous on Power Flow at Interval
[✓ Constant] hr h min m sec s ms ms
Variable D0 ms

On Success: [✓ Set bit] JMP to Stage C0
On Success Counter D0
On Error: [✓ Set bit] JMP to Stage C1
On Error Counter D1
Extended Error Information D2

Row#	Read From Remote	Into Local
1		

Size: 0 bytes Bytes Remaining: 1000 bytes

RX, continued

IP Address – The IP Address of the Do-more! CPU to read the data from. This can be either a Fixed (static) IP Address or a Variable (dynamic) value.

Fixed IP Address – The TCP Address assigned to the Do-more! CPU. IP addresses are canonically represented in dot-decimal notation, consisting of four decimal numbers, each ranging from 0 to 255, separated by dots.

Variable IP Address – The IP Address resides in the specified memory location. This can be any readable DWord numeric location.

UDP Port Number – The port number of the Do-more! CPU) to read the data from. The default value of 28784 (0x7070) is typically the correct number for Do-more! protocol. It is possible to use a different UDP Port Number if the Enable Secondary Ethernet Connection is turned on under the CPU Configuration settings. This secondary UDP Port Number defaults to 5000. This UDP Port Number can be any decimal value between 5000 and 65535, except for 28784 (the port number used by Do-more Designer)..

Remote Password – The RX instruction requires that a communication session be established with the remote Do-more! CPU before the data read operations can be processed. Depending on how the remote system is configured, this may require the user to enter the password for the User Account to allow the session to be established. If no remote password is selected, the connection will be established using the Default User account.

Enable – Designates how this instruction is enabled. Select from one of the following:

Once on Leading Edge – Select this option to have this instruction run to completion exactly one time. Typically, this will take more than one controller scan. Configured this way the Do-more! Network Read (RX) instruction is Edge Triggered.

Continuous on Power Flow at Interval – Select this option to have this instruction run as long as the instruction has power flow. After the Do-more! Network Read (RX) has initially run, if the instruction still has power flow, the instruction will remain enabled and will wait the specified amount of time before running again. The following options select how much time (in milliseconds) to wait between successive runs.

- **Constant** – specifies the interval time in Hours / Minutes / Seconds / Milliseconds.
- **Variable** – This can be any readable numeric location that contains a value between 0 and 2,147,483,647 which represents the number of milliseconds to wait before running again. A value of 0 ms means this instruction will be set to run on the next scan.

On Success – Selects which of the following actions to perform if the Network Read operation is successful:

Set Bit – Enable this selection then specify any writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

On Success Counter – Enable this option and select a DWord location to store the total number of times the RX completed successfully. This can be any DWord location.

On Error – Selects which of the following actions to perform if the Network Read operation

RX, continued

is unsuccessful:

Set Bit – Enable this selection then specify writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

On Error Counter – Enable this option and select a DWord location to store the total number of times the RX failed to complete. This can be any DWord location.

Extended Error Information – Enable this selection then enter a memory location to store any error codes returned for this instruction. This can be any writable numeric location. The list following is of the Extended error responses and their meaning:

-1 = Protocol Error occurred. The value in LastProtoError (DST38) contains the protocol error code as follows:

2 (0x02) = Out Of Sessions: the remote PLC currently has 32 concurrent connections and cannot accept any more.

3 (0x03) = Illegal Operation: the User Account for the password in the instruction does not have Write Data privilege.

4 (0x04) = Invalid Session: an error occurred with the session to the remote PLC, for example the remote PLC lost power during the session.

6 (0x06) = Invalid Argument: the write requests are not formed properly (you should never see this).

14 (0x0E) = Invalid Password: the password in the instruction does not match any User Account in the remote PLC,

20 (0x14) = Bad DMPP Request: one or more of the write requests cannot be processed, most likely the request is for a location that is out of range on the remote PLC or the memory block doesn't exist on the remote PLC. The Extended Error location will contain the entry number of the write request that is causing the error.

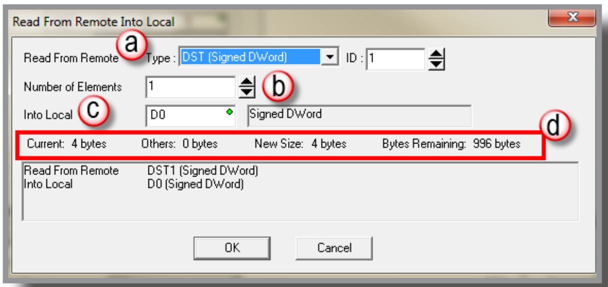
0 = No Extended Error

1 – 50 = row number of the bad read request if the Last Protocol Error Code was 20 (0x14).

Insert – Select Insert to open ***Read From Remote Into Local*** setup dialog box (See page following).

RX, continued

Read From Remote Into Local



Read From Remote Type : DST (Signed DWord) ID : 1
Number of Elements 1
Into Local D0 Signed DWord
Current: 4 bytes Others: 0 bytes New Size: 4 bytes Bytes Remaining: 996 bytes

```
Read From Remote    DST1 (Signed DWord)
Into Local          D0 (Signed DWord)
```

Read From Remote – The first location in the remote PLC to begin reading data:

Type (a) – The Block Name of the remote memory location to read. This drop-down list contains an entry for all of the built-in memory data blocks, all of the built-in structures, and an entry for a User Block that exists on the remote PLC.

Number of Elements (b) – The number of consecutive elements to read.

Into Local (c) – The beginning location in the PLC to store the data from the read operation. The Into Local location must be the same data type and have the same element size as the specified Read From Remote data location. For example you can read from WY (Signed Word) into N (Signed Word) but you cannot read from WY (Signed Word) into V (Unsigned Word).

The remaining fields **(d)** are to help you maintain the size limitations. Fields validate immediately as you make changes in the editor.

Current – The number of bytes in the read request currently being edited.

Others – The number of bytes in the read requests already in this instruction.

New Size – The total number of bytes of all the read requests in the instructions, including the one currently being edited.

Bytes Remaining – The number of bytes of the 1000 byte limit remaining for read requests.

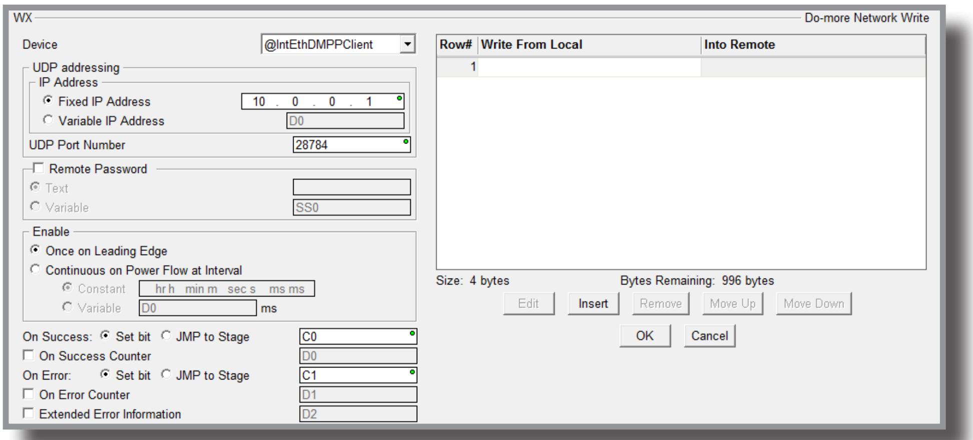
WX

The Do-more! Network Write (WX) instruction uses the Do-more! proprietary protocol to write data over the on-board Ethernet port to another Ethernet-equipped Do-more! CPU. WX uses UDP (not TCP) protocol to communicate with the remote Do-more! CPU.

Each WX instruction can contain up to 50 individual write requests for a total of up to 1000 bytes of data. The WX instruction can write to all of the built-in memory blocks, all of the built-in structures, and any user-created memory blocks in the remote PLC. WX does NOT support writing to a Heap Item in the remote Do-more! CPU.

In contrast to DirectLOGIC (DLRX / DLWX) and Modbus/TCP (MRX / MWX) network communication which only has access to the protocol-specific memory blocks in the remote CPUs, Do-more! WX has direct access to nearly all of the memory in the remote Do-more! CPU – including direct access to the PLC's I/O memory.

The WX instruction establishes a session with the remote Do-more! CPU. The session is established using one of the User Accounts on the remote Do-more! CPU. It is through that User Account (System Security) that any access restrictions on what can be accessed can be enforced. By default it will use the Default User account (no password required).


WX — Do-more Network Write

Device: @IntEthDMPPClient

UDP addressing
IP Address
(● Fixed IP Address) 10 . 0 . 0 . 1
( ) Variable IP Address D0
UDP Port Number 28784

☐ Remote Password
(● Text)
( ) Variable SS0

Enable
(● Once on Leading Edge)
( ) Continuous on Power Flow at Interval
(● Constant) hr h min m sec s ms ms
( ) Variable D0 ms

On Success: (● Set bit) ( ) JMP to Stage C0
☐ On Success Counter D0
On Error: (● Set bit) ( ) JMP to Stage C1
☐ On Error Counter D1
☐ Extended Error Information D2

Row#	Write From Local	Into Remote
1		

Size: 4 bytes Bytes Remaining: 996 bytes

IP Address – The IP Address of the Do-more! CPU to write the data to. This can be either a Fixed (static) IP Address or a Variable (dynamic) value.

Fixed IP Address – The TCP Address assigned to the Do-more! CPU. IP addresses are canonically represented in dot-decimal notation, consisting of four decimal numbers, each ranging from 0 to 255, separated by dots.

Variable IP Address – The IP Address resides in the specified memory location. This can be any readable DWord numeric location.

UDP Port Number – The port number of the Do-more! CPU) to write the data to. The default value of 28784 (0x7070) is typically the correct number for Do-more! protocol. It is possible to use a different UDP Port Number if the Enable Secondary Ethernet Connection is turned on under the CPU Configuration settings. This secondary UDP Port Number defaults

WX, continued

to 5000. This UDP Port Number can be any decimal value between 5000 and 65535, except for 28784 (the port number used by Do-more Designer).

Remote Password – The WX instruction requires that a communication session be established with the remote Do-more! CPU before the data write operations can be processed. Depending on how the remote system is configured, this may require the user to enter the password for the User Account to allow the session to be established. If no remote password is selected, the connection will be established using the Default User account.

Enable – Designates how this instruction is enabled. Select from one of the following:

Once on Leading Edge – Select this option to have this instruction run to completion exactly one time. Typically, this will take more than one controller scan. Configured this way the Do-more Network Write (WX) instruction is Edge Triggered.

Continuous on Power Flow at Interval – select this option to have this instruction run as long as the instruction has power flow. After the Do-more! Network Write (WX) has initially run, if the instruction still has power flow, the instruction will remain enabled and will wait the specified amount of time before running again. The following options select the time (in milliseconds) delay between successive runs.

Constant - specifies the interval time in Hours / Minutes / Seconds / Milliseconds.

Variable - This can be any readable numeric location that contains a value between 0 and 2,147,483,647 which represents the number of milliseconds to wait before running again. A value of 0 ms means this instruction will be set to run on the next scan.

On Success – Selects which of the following actions to perform if the Network Write operation is successful:

Set Bit – Enable this selection then specify any writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

On Success Counter – Enable this option and select a DWord location to store the total number of times the WX completed successfully. This can be any DWord location.

WX, continued

On Error – Selects which of the following actions to perform if the Network Write operation is unsuccessful:

Set Bit – Enable this selection then specify writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

On Error Counter – Enable this option and select a DWord location to store the total number of times the RX failed to complete. This can be and DWord location.

Extended Error Information – Enable this selection then enter a memory location to store any error codes returned for this instruction. This can be any writable numeric location. The following lists the Extended error responses and their meaning:

-1 = Protocol Error occurred. The value in LastProtoError (DST38) contains the protocol error code as follows:

2 (0x02) = Out Of Sessions: the remote PLC currently has 32 concurrent connections and cannot accept any more.

3 (0x03) = Illegal Operation: the User Account for the password in the instruction does not have Write Data privilege.

4 (0x04) = Invalid Session: an error occurred with the session to the remote PLC, for example the remote PLC lost power during the session.

6 (0x06) = Invalid Argument: the write requests are not formed properly (you should never see this).

14 (0x0E) = Invalid Password: the password in the instruction does not match any User Account in the remote PLC.

20 (0x14) = Bad DMPP Request: one or more of the write requests cannot be processed, most likely the request is for a location that is out of range on the remote PLC or the memory block doesn't exist on the remote PLC. The Extended Error location will contain the entry number of the write request that is causing the error.

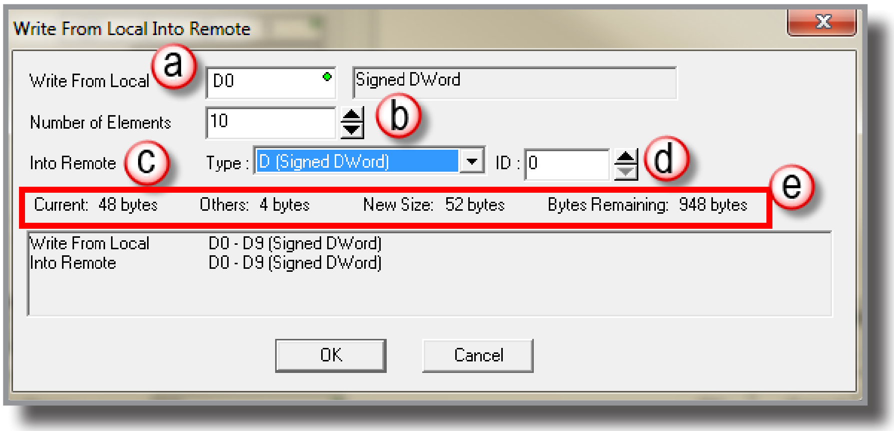
0 = No Extended Error

1 – 50 = Row number of the bad read request if the Last Protocol Error Code was 20 (0x14).

Insert – Select Insert to open Write From Local Into Remote dialog box.

WX, continued

Write From Local Into Remote



Write From Local: D0 — Signed DWord
Number of Elements: 10
Into Remote: Type : D (Signed DWord) ID : 0

Current: 48 bytes Others: 4 bytes New Size: 52 bytes Bytes Remaining: 948 bytes

```
Write From Local      D0 - D9 (Signed DWord)
Into Remote           D0 - D9 (Signed DWord)
```

Write from Local (a) – The first location of the source data in the Local PLC:

Number of Elements (b) – The number of consecutive elements to write.

Into Remote (c) – The beginning location in the remote PLC to store the data from the write operation. The Into Remote location must be the same data type and have the same element size as the specified Write From Local data location. For example you can write from WY (Signed Word) into N (Signed Word) but you cannot write from WY (Signed Word) into V (Unsigned Word).

ID (d) – The beginning offset to write the data.

The remaining fields (e) are to help you maintain the size limitations. These fields validate immediately as you make changes in the editor.

Current – The number of bytes in the read request currently being edited.

Others – The number of bytes in the read requests already in this instruction.

New Size – The total number of bytes of all the read requests in the instructions, including the one currently being edited.

Bytes Remaining – The number of bytes of the 1000 byte limit remaining for read requests.

Modbus TCP/IP

Modbus TCP is a protocol overseen by Modbus.org. This standard is an open standard meaning that anyone can utilize it freely.

Modbus TCP can be utilized as either a client or server configuration. It supports Clients and Servers in a Peer to Peer fashion.

Server

As a Modbus TCP Server (Slave), the BRX Do-more MPU is functioning as a listening/replying device. The external Client (Master) device will request data registers from the BRX Do-more! and the BRX Do-more! will reply with the appropriate data.

All Modbus Client data is stored in four sets of registers in the BRX Do-more!. This memory area is blocked off specifically for Modbus communications. You must place data in these registers in order for a Modbus Client device to be able to access it.

Modbus Data Registers

Register Type	Register Name	Range
Holding Coil	MC	00000–01023
Input Coil	MI	10000–11023
Holding Register	MHR	30000–32047
Input Register	MIR	40000–42047

The Modbus data area is loosely data typed and casting or other instructions such as PUBLISH and SUBSCRIBE can be utilized to convert data in this area to the proper data type needed as well. Please see the help file for more information on casting, PUBLISH and SUBSCRIBE.

NOTE: *Ranges can be expanded in the Memory Configuration section of the Do-more! Designer software as needed.*

The Modbus TCP Server (Slave) supports the following function codes:

Function Code	Description
1	Read Coils
2	Read Discrete Inputs
3	Read Holding Registers
4	Read Input Registers
5	Write Single Coils
6	Write Single Registers
7	Read Exception Status
15	Write Multiple Coils
16	Write Multiple Registers
22	Mask Write Register

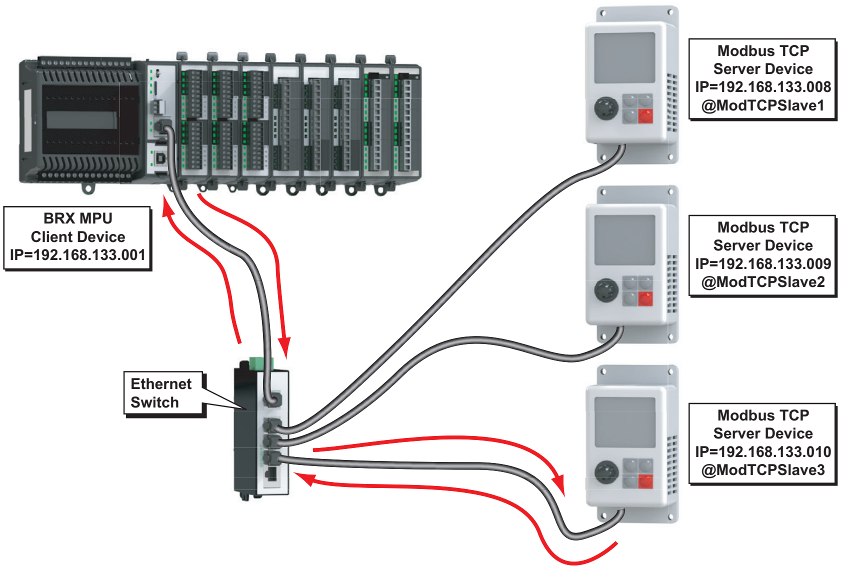
Modbus TCP Client (Master)

As a Modbus TCP Client (Master), the BRX Do-more! MPU is requesting data from a Modbus TCP Server (Slave) device.

In order for this to work, you need to know quite a few things about your Server (Slave) device such as the function codes that it supports, the data registers that are accessible and possibly the Unit ID or Slave Address.

In order to utilize the BRX Do-more! as a Modbus TCP Client you will need to create a Modbus TCP Client Device. This device will handle all of the communications with the external Modbus servers by using the Ethernet port on the front of the BRX MPU.

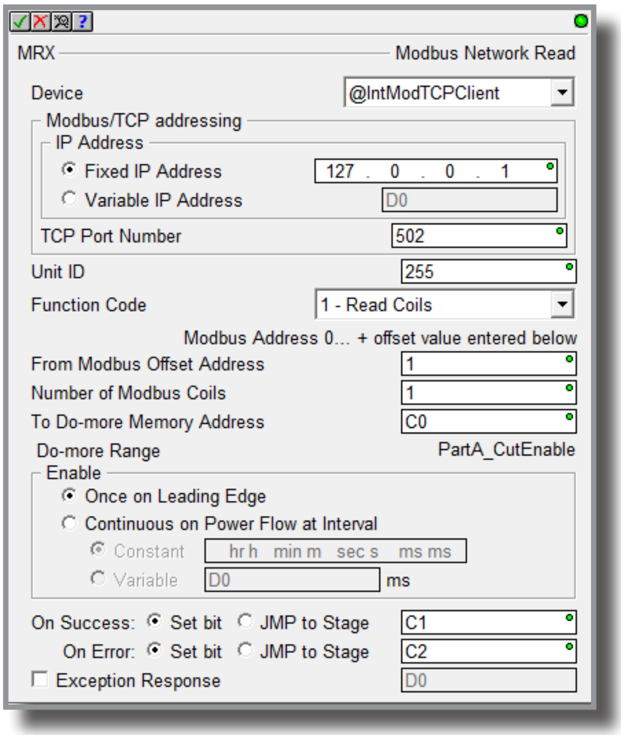
Note that there is a default @IntMODTCPClient device created automatically. You can choose to use this device, however we strongly recommend that you make a new Device for each Modbus TCP server that you will be communicating with. This will allow simultaneous communications to flow uninterrupted if one of the Modbus TCP servers goes offline.



As a Modbus TCP Client, you do not have to sequence the MRX read and MRX write instructions. If you are so inclined, you can just drop them into your program and they will work in a round robin manner. However, sequencing the instructions will give you better control and allow you to build complex communication patterns to optimally communicate with your devices.

The communications instructions in Do-more! Designer have Success and Error built into the instructions so that you can either set a bit or move to a Stage if you are doing state style programming. There are examples of using Stage (state) programming in the software help file to show how you could use this to implement a complex communications routine.

MRX Instruction



MRX — Modbus Network Read

Device	@IntModTCPClient
Modbus/TCP addressing	
IP Address	
● Fixed IP Address	127 . 0 . 0 . 1
○ Variable IP Address	D0
TCP Port Number	502
Unit ID	255
Function Code	1 - Read Coils
	Modbus Address 0... + offset value entered below
From Modbus Offset Address	1
Number of Modbus Coils	1
To Do-more Memory Address	C0
Do-more Range	PartA_CutEnable
Enable	
● Once on Leading Edge	
○ Continuous on Power Flow at Interval	
● Constant	hr h min m sec s ms ms
○ Variable	D0 ms
On Success: ● Set bit ○ JMP to Stage	C1
On Error: ● Set bit ○ JMP to Stage	C2
☐ Exception Response	D0

The MRX instruction is used to read from a Modbus TCP Server. Following is a brief description of the MRX instruction parameters. For more detailed information please refer to the Do-more! Designer help files.

Device – The device associated with the physical port that you want to communicate from. @IntModTCPClient is the name of the device associated with the built in Ethernet port when set as a Modbus TCP Client. We strongly recommend that you make a new Device for each Modbus TCP server that you will be communicating with. This will allow simultaneous communications to flow uninterrupted if one of the Modbus TCP servers goes offline.

IP Address – The IP Address of the Modbus TCP Server (Slave) to read the data from. This can be either a Fixed (static) IP Address or a Variable (dynamic) value.

Fixed IP Address – The TCP Address assigned to the Modbus TCP Server (Slave). IP addresses are canonically represented in dot-decimal notation, consisting of four decimal numbers, each ranging from 0 to 255, separated by dots.

Variable IP Address – The IP Address resides in the specified memory location. This can be any readable DWord numeric location.

TCP Port Number – This should normally be set to 502 unless the end device has had the default port number changed.

Unit ID – The ID number of the Server (Slave) device. Typically this is 255 unless you are talking to a Modbus Serial Gateway style of device.

Function Code – selects which of the following Modbus function codes to use:

1 - Read Coils

2 - Read Discrete Inputs

3 - Read Holding Registers

4 - Read Input Registers

7 - Read Exception Status

MRX Instruction, continued

From Modbus Offset Address – The address in the Modbus Server (Slave) that you will be reading from. This address may be offset by a value of +1 depending on how the manufacturer followed the Modbus standard.

Number of Modbus Coils/ Registers – Based on the Function Code selected, this selection specifies how many consecutive elements to read.

To Do-more Memory Address – Specifies the beginning address of a range of bits or numeric locations in the CPU where the data that is read will be stored. This data type (bit or register) must match the type expected by the Function Code.

Do-more Range – This is the ending register where the data will be stored at, calculated by taking the **To Do-more Memory** address and adding the **Number of Modbus Coils/Registers** value to it.

Enable – Designates how this instruction will operate. Select from one of the following:

Once on Leading Edge - Select this option to have this instruction run to completion exactly one time. Typically, this instruction will take more than one controller scan to complete. Configured this way the instruction is Edge Triggered.

Continuous on Power Flow at Interval - Select this option to have this instruction run as long as the instruction has power flow. After the instruction has initially run, if the instruction still has power flow, the instruction will remain enabled and will wait the specified amount of time before running again. The following options select how much time (in milliseconds) to wait between successive runs. A value of 0ms means the instruction will re-run immediately.

On Success – When the instruction completes successfully this action will be performed.

Set Bit – Enable this selection then specify any writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

On Error – When the instruction does not complete successfully this action will be performed.

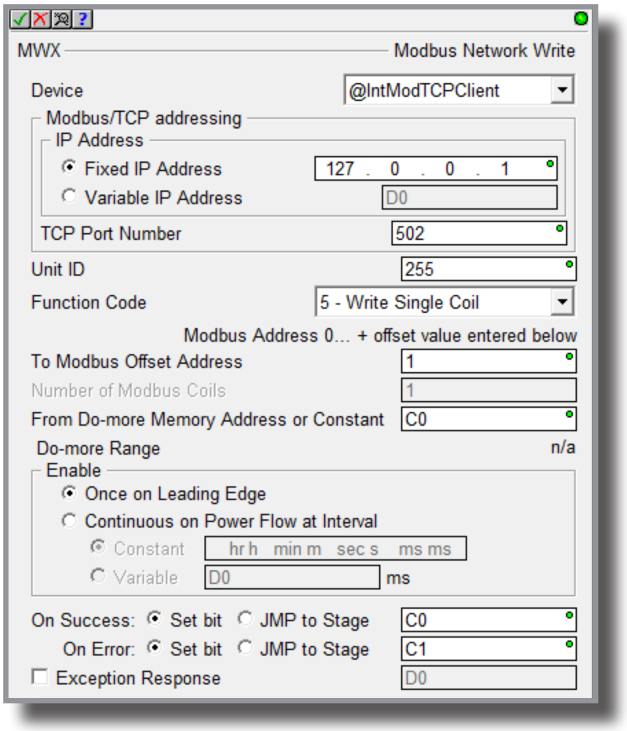
Set Bit – Enable this selection then specify any writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

Exception Response – For errors where the message was received properly by the Server (Slave) device, this will contain a value to indicate why the Server rejected the message. This can aid in troubleshooting the issue with the message.

MWX Instruction

The MWX instruction is used to write to a Modbus TCP Server. For specific information please refer to the Do-more! Designer help files.



MWX — Modbus Network Write

Device: @IntModTCPClient

Modbus/TCP addressing

IP Address

[✓ Fixed IP Address] 127 . 0 . 0 . 1

Variable IP Address D0

TCP Port Number: 502

Unit ID: 255

Function Code: 5 - Write Single Coil

Modbus Address 0... + offset value entered below

To Modbus Offset Address: 1

Number of Modbus Coils: 1

From Do-more Memory Address or Constant: C0

Do-more Range: n/a

Enable

[✓ Once on Leading Edge]

Continuous on Power Flow at Interval

[✓ Constant] hr h min m sec s ms ms

Variable D0 ms

On Success: [✓ Set bit] JMP to Stage C0

On Error: [✓ Set bit] JMP to Stage C1

Exception Response D0

Device – The device associated with the physical port that you want to communicate from. @IntModTCPClient is the name of the device associated with the built in Ethernet port when set as a Modbus TCP Client. We strongly recommend that you make a new Device for each Modbus TCP Server (Slave) that you will be communicating with. This will allow simultaneous communications to flow uninterrupted if one of the Modbus TCP servers goes offline.

IP Address – The IP Address of the Modbus TCP Server (Slave) to read the data from. This can be either a Fixed (static) IP Address or a Variable (dynamic) value.

Fixed IP Address – The TCP Address assigned to the Modbus TCP Server (Slave). IP addresses are canonically represented in dot-decimal notation, consisting of four decimal numbers, each ranging from 0 to 255, separated by dots.

Variable IP Address – The IP Address resides in the specified memory location. This can be any readable DWord numeric location.

MWX Instruction, continued

TCP Port Number – This should normally be set to 502 unless the end device has had the default port number changed.

Unit ID – The ID number of the Modbus TCP Server (Slave) device. Typically this is 255 unless you are talking to a Modbus Serial Gateway style of device.

Function Code – Selects which of the following Modbus function codes to use:

5 - Write Single Coil

6 - Write Single Register

15 - Write Multiple Coils

16 - Write Multiple Registers

To Modbus Offset Address – The starting register that you will be writing to. This address may be offset by a value of +1 depending on how the manufacturer followed the Modbus standard.

Number of Modbus Coils/ Registers – This selection specifies how many consecutive elements to write from the **Modbus Offset Address**.

From Do-more Memory Address – Specifies the beginning address of a range of bits or numeric locations in the CPU where the data that will be written from. This data type (bit or register) must match the type expected by the Function Code.

Do-more Range – This is the ending register where the data will be written from, calculated by taking the To Do-more Memory address and adding the Number of Modbus Coils/Registers value to it.

Enable – Designates how this instruction will operate. Select from one of the following:

Once on Leading Edge – Select this option to have this instruction run to completion exactly one time. Typically, this instruction will take more than one controller scan to complete. Configured this way the instruction is Edge Triggered.

Continuous on Power Flow at Interval – Select this option to have this instruction run as long as the instruction has power flow. After the instruction has initially run, if the instruction still has Power Flow, the instruction will remain enabled and will wait the specified amount of time before running again. The following options select how much time (milliseconds) to wait between successive runs. A value of 0ms means the instruction will re-run immediately.

On Success – When the instruction completes successfully this action will be performed.

Set Bit – Enable this selection then specify any writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

On Error – When the instruction does not complete successfully this action will be performed.

Set Bit – Enable this selection then specify any writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

Exception Response – For errors where the message was received properly by the Server (Slave) device, this will contain a value to indicate why the Server rejected the message. This can aid in troubleshooting the issue with the message.

HOST Ethernet Protocol

Server

The BRX Do-more! MPU can serve as a HOST Ethernet Protocol server to communicate to legacy devices that utilize the HOST Ethernet protocol such as DirectLogic PLC's, C-more HMI, SCADA systems, etc.

All data is stored in four sets of registers in the BRX Do-more!. This memory area is blocked off specifically for HOST Ethernet Protocol communications. You must place data in these registers so that a HOST Ethernet Protocol Client device will be able to access it.

HOST Data Registers

DirectLogic Type	Do-more Block Name	Default Octal Range
Discrete Input (X)	DLX	0–777
Discrete Output (Y)	DLY	0–777
Control Relay (C)	DLC	0–777
16-Bit Data (V)	DLV	0–3777

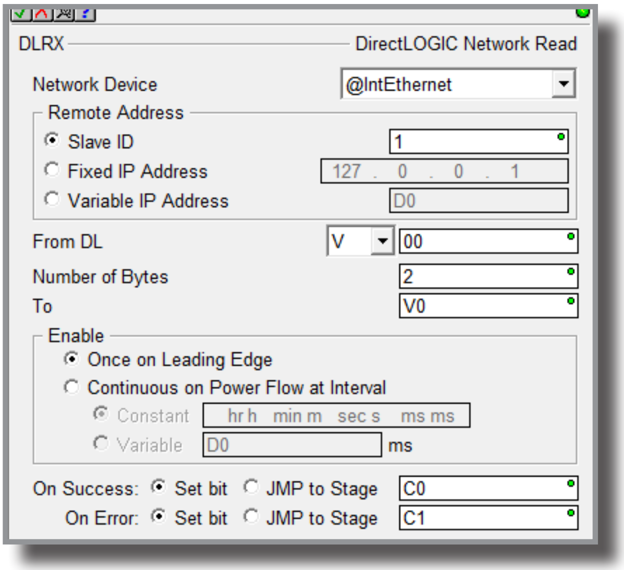
The HOST Ethernet Protocol data area is loosely data typed and casting or other instructions such as PUBLISH and SUBSCRIBE can be utilized to convert data in this area to the proper data type needed as well. Please see the help file for more information on casting, PUBLISH and SUBSCRIBE.

NOTE: *Ranges can be expanded as needed in the Memory Configuration section of the Do-more! Designer software.*

DirectLogic Client (Master)

The BRX Do-more! MPU can be a HOST Ethernet Protocol Client to communicate to legacy devices that utilize the HOST Ethernet protocol such as DirectLogic PLC's.

DLRX



DLRX — DirectLOGIC Network Read

Network Device: @IntEthernet

Remote Address
- (● Slave ID) 1
- (Fixed IP Address) 127 . 0 . 0 . 1
- (Variable IP Address) D0

From DL: V 00

Number of Bytes: 2

To: V0

Enable
- (● Once on Leading Edge)
- (Continuous on Power Flow at Interval)
  - (● Constant) hr h min m sec s ms ms
  - (Variable) D0 ms

On Success: [✓ Set bit] JMP to Stage C0

On Error: [✓ Set bit] JMP to Stage C1

Network Device – The device associated with the physical port that you want to communicate from. @IntEthernet is the name of the device associated with the built in Ethernet port.

Remote Address – specifies which of the following addressing modes to use:

Slave ID - Selects the Slave ID (Module ID) of the remote DirectLOGIC slave. The Slave ID can be any constant value in the range of 1 to 90, or any readable numeric location that contains a value in that range. If Slave ID is selected, a TCP/IP broadcast is used to perform the network read operation. This means that both the Do-more! controller and the remote ECOM module must be in the same Broadcast Domain.

Fixed IP Address – Specifies the IP Address assigned to the remote slave ECOM module. IP addresses are canonically represented in dot-decimal notation, consisting of four decimal numbers, each ranging from 0 to 255, separated by dots.

Variable IP Address – The IP Address resides in a memory location in the PLC, allowing the IP Address to be changed at runtime. This can be any readable DWord numeric location. Each octet of the IP Address is stored in one byte of the Variable Address location.

DLRX Instruction, continued

From DL - Designates the data type and the address of the data to read from the DirectLOGIC controller.

V-Memory – Locations in a DirectLOGIC controller are unsigned 16-bit values. Each V-Memory location is 2 bytes in length, so reading V-Memory requires the length be in 2-byte increments. The memory address value must begin on a byte boundary. Single Bit locations in the DirectLOGIC controllers cannot be read individually, you must read the byte that contains the desired bit.

Number of Bytes – Designates the number of elements of the selected type to read.
X, Y, C, S, T, CT, GX, GY, SP – Bit locations must be read in 1-byte increments

Enable – designates how this instruction will operate. Select from one of the following:

Once on Leading Edge - select this option to have this instruction run to completion exactly one time. Typically, this instruction will take more than one controller scan to complete. Configured this way the instruction is Edge Triggered.

Continuous on Power Flow at Interval - Select this option to have this instruction run as long as the instruction has power flow. After the instruction has initially run, if the instruction still has power flow, the instruction will remain enabled and will wait the specified amount of time before running again. The following options select how much time (in milliseconds) to wait between successive runs. A value of 0ms means the instruction will re-run immediately.

On Success – When the instruction completes successfully this action will be performed.

Set Bit – Enable this selection then specify any writable bit location.

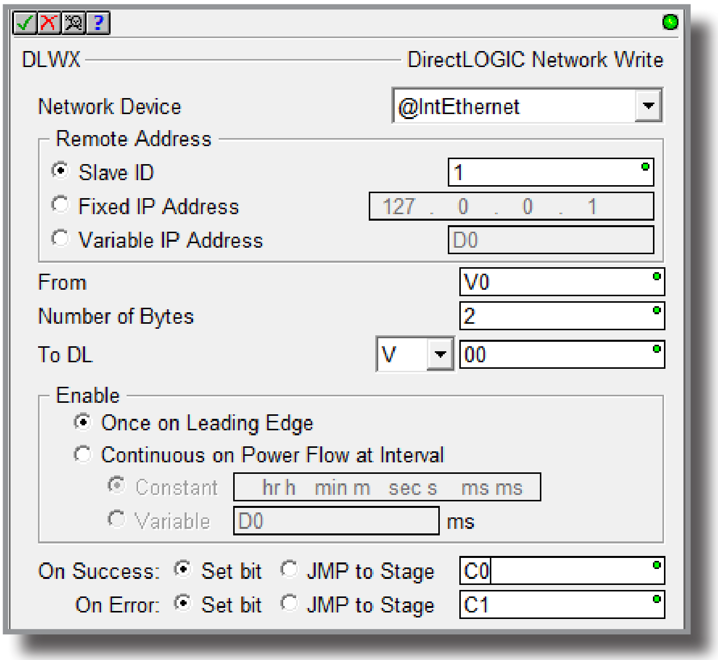
JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

On Error – When the instruction does not complete successfully this action will be performed.

Set Bit – Enable this selection then specify any writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

DLWX



DLWX — DirectLOGIC Network Write

Network Device: @IntEthernet

Remote Address
- (● Slave ID) 1
- (Fixed IP Address) 127 . 0 . 0 . 1
- (Variable IP Address) D0

From: V0

Number of Bytes: 2

To DL: V 00

Enable
- (● Once on Leading Edge)
- (Continuous on Power Flow at Interval)
  - (● Constant) hr h min m sec s ms ms
  - (Variable) D0 ms

On Success: (● Set bit) (JMP to Stage) C0

On Error: (● Set bit) (JMP to Stage) C1

Network Device – The device associated with the physical port that you want to communicate from. **@IntEthernet** is the name of the device associated with the built in Ethernet port.

Remote Address - Specifies which of the following addressing modes to use:

Slave ID – Selects the Slave ID (Module ID) of the remote DirectLOGIC slave. This can be any constant value in the range of 1 to 90, or any readable numeric location that contains a value in that range. If Slave ID is selected, the a TCP/IP broadcast is used to perform the network read operation, this means that both the Do-more controller and the remote ECOM module must be in the same Broadcast Domain.

Fixed IP Address – Specifies the IP Address assigned to the remote slave ECOM module. IP addresses are canonically represented in dot-decimal notation, consisting of four decimal numbers, each ranging from 0 to 255, separated by dots.

Variable IP Address – The IP Address resides in a memory location in the PLC. This allows the IP Address to be changed at runtime. This can be any readable DWord numeric location. Each octet of the IP Address is stored in one byte of the Variable Address location.

DLWX Instruction, continued

From – Specifies the beginning memory address in the Do-more controller of the data to send to the remote slave. This value can be any readable numeric location.

To DL – Designates the data type and the address of the data to Write to the DirectLOGIC controller.

V-Memory locations in a DirectLOGIC controller are unsigned 16-bit values. Each V-Memory location is 2 bytes in length, so reading V-Memory requires the length be in 2-Byte increments. The memory address value must begin on a Byte boundary. Single Bit locations in the DirectLOGIC controllers cannot be read individually, you must read the Byte that contains the desired Bit.

Number of Bytes – Designates the number of elements of the selected type to read.
X, Y, C, S, T, CT, GX, GY, SP – Bit locations must be read in 1-Byte increments.

Enable – designates how this instruction will operate. Select from one of the following:

Once on Leading Edge – Select this option to have this instruction run to completion exactly one time. Typically, this instruction will take more than one controller scan to complete. Configured this way the instruction is Edge Triggered.

Continuous on Power Flow at Interval – Select this option to have this instruction run as long as the instruction has power flow. After the instruction has initially run, if the instruction still has **Power Flow**, the instruction will remain enabled and will wait the specified amount of time before running again. The following options select how much time (in milliseconds) to wait between successive runs. A value of 0ms means the instruction will re-run immediately.

On Success – When the instruction completes successfully this action will be performed.

Set Bit – Enable this selection then specify any writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

On Error – When the instruction does not complete successfully this action will be performed.

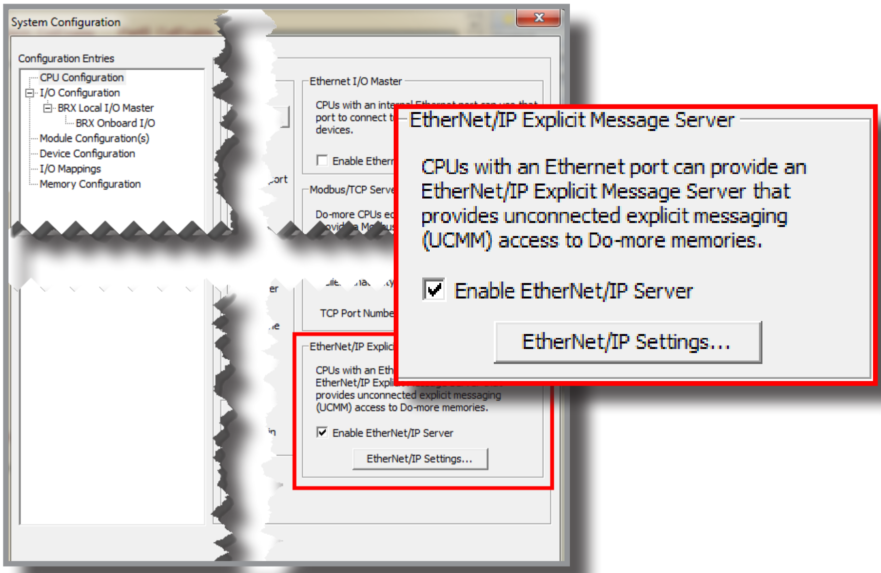
Set Bit – Enable this selection then specify any writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

EtherNet/IP (Explicit Messaging)

Server

The EtherNet/IP Explicit Message Server must be enabled in the *System Configuration* dialog box.



System Configuration

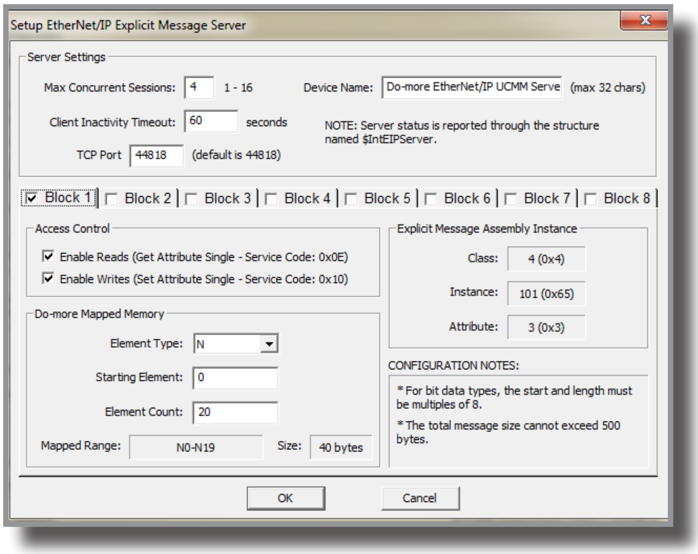
Configuration Entries
- CPU Configuration
- I/O Configuration
  - BRX Local I/O Master
    - BRX Onboard I/O
- Module Configuration(s)
- Device Configuration
- I/O Mappings
- Memory Configuration

EtherNet/IP Explicit Message Server

CPUs with an Ethernet port can provide an EtherNet/IP Explicit Message Server that provides unconnected explicit messaging (UCMM) access to Do-more memories.

☑ Enable EtherNet/IP Server

Once this is enabled, you must click the EtherNet/IP Settings button to set the data blocks that will be served to outside EtherNet/IP Clients.



Setup EtherNet/IP Explicit Message Server

Server Settings

Max Concurrent Sessions: 4 1 - 16

Device Name: Do-more EtherNet/IP UCMM Serve (max 32 chars)

Client Inactivity Timeout: 60 seconds

NOTE: Server status is reported through the structure named $IntEIPServer.

TCP Port 44818 (default is 44818)

☑ Block 1 | ☐ Block 2 | ☐ Block 3 | ☐ Block 4 | ☐ Block 5 | ☐ Block 6 | ☐ Block 7 | ☐ Block 8

Access Control

☑ Enable Reads (Get Attribute Single - Service Code: 0x0E)

☑ Enable Writes (Set Attribute Single - Service Code: 0x10)

Explicit Message Assembly Instance

Class: 4 (0x4)

Instance: 101 (0x65)

Attribute: 3 (0x3)

Do-more Mapped Memory

Element Type: N

Starting Element: 0

Element Count: 20

Mapped Range: N0-N19 Size: 40 bytes

CONFIGURATION NOTES:

* For bit data types, the start and length must be multiples of 8.

* The total message size cannot exceed 500 bytes.

EtherNet/IP (Explicit Messaging), continued

TCP Port Number (44818 is default) – Designates the TCP port number on which the EtherNet/IP Explicit Message Server will accept connections. The default value of 44818 is the industry standard and will rarely need to be changed. However, this can be any constant value between 0 and 65535.

Device Name – Up to 32 characters that will be returned in requests for the Identity Class.

NOTE: Runtime status of the EtherNet/IP Server is accessed through the built-in structure $IntEIPServer which has the following members:

.ActiveSessions (read-only) – The number of concurrent open connections to EtherNet/IP clients.

.LastError – Last error reported to an EtherNet/IP client.

.Errors – Total number of errors returned to all EtherNet/IP clients.

.Transactions – Total number or completed client requests to EtherNet/IP clients.

Select the quantity of Data Blocks (up to 8 data blocks - Block 1 to Block 8) that will be made available to EtherNet/IP Clients. You can specify how many data blocks will be used and then configure each of the blocks in the sections below.

Access Control – Specify external access ability.

Enable Reads (Get Single Attribute – Service Code: 0x0E) – Allows EtherNet/IP Clients to read from this data block using Get Single Attribute.

Enable Writes (Set Single Attribute – Service Code: 0x10) – Allows EtherNet/IP Clients to write to this data block using Set Single Attribute.

Do-more Mapped Memory – Specifies the first location in a data block in the Do-more PLC memory that will be accessed when requests with this Class/Instance/Attribute are received.

Element Type – Select the memory block to use from the drop-down list of available numeric memory blocks.

Starting Element – Specify the first element in the memory block to use.

Element Count – The number of successive Elements in the Do-more memory block to use. The maximum total size on an EtherNet/IP request is 500 bytes, so the maximum number of elements per block will depend on the size of the individual elements. Refer to the chart below:

Mapped Range – Displays the currently selected range of Elements.

Size – Displays the size of the selected Element range in Bytes.

Explicit Messaging Assembly Instance – Displays the Path (class / instance / attribute) of the data block being configured.

Element Count

Element Type	Maximum Number of This Type per Data Block
Bit	4000
Byte	500
Word	250
DWord	125
Real	125

EtherNet/IP (Explicit Messaging), continued

Class – The Class of all the data blocks is fixed at 0x04 (assembly class).

Instance – Each of the 8 blocks is assigned a unique Instance as shown in this table.

Attribute – The Attribute of each of these data blocks is fixed at 0x03.

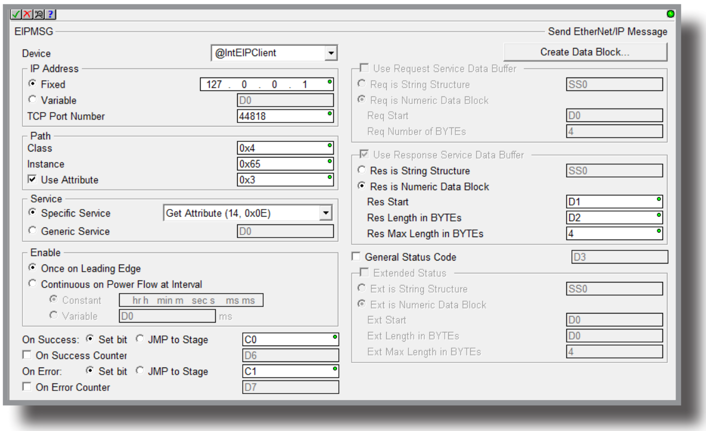
CONFIGURATION NOTES: – This area contains information pertinent to the configuration selections being made.

Instance Blocks

Block Number	Instance ID
0	101 (0x65)
1	102 (0x66)
2	103 (0x67)
3	104 (0x68)
4	105 (0x69)
5	106 (0x6A)
6	107 (0x6B)
7	108 (0x6C)

EtherNet/IP Client (Master)

EIPMSG Instruction – The Send EtherNet/IP Message instruction implements an Explicit Unconnected EtherNet/IP Client using the on-board Ethernet port of a Do-more! CPU. An explicit message client initiates request/response oriented communications with EtherNet/IP servers. Message rates and latency requirements should not be too demanding. Examples of other explicit message servers you can talk to are barcode scanners, scales, drives, or other intelligent devices.



EIPMSG — Send EtherNet/IP Message

Device: @IntEIPClient — Create Data Block...

IP Address: [✓ Fixed] 127 . 0 . 0 . 1 / Variable D0
TCP Port Number: 44818

Path
Class: 0x4
Instance: 0x65
[✓ Use Attribute] 0x3

Service
[✓ Specific Service] Get Attribute (14, 0x0E)
Generic Service D0

Enable
[✓ Once on Leading Edge]
Continuous on Power Flow at Interval
Constant hr h min m sec s ms ms
Variable D0 ms

On Success: [✓ Set bit] JMP to Stage — C0
On Success Counter — D6
On Error: [✓ Set bit] JMP to Stage — C1
On Error Counter — D7

Use Request Service Data Buffer
Req is String Structure — SS0
[✓ Req is Numeric Data Block]
Req Start — D0
Req Number of BYTEs — 4

[✓ Use Response Service Data Buffer]
Res is String Structure — SS0
[✓ Res is Numeric Data Block]
Res Start — D1
Res Length in BYTEs — D2
Res Max Length in BYTEs — 4

General Status Code — D3
Extended Status
Ext is String Structure — SS0
[✓ Ext is Numeric Data Block]
Ext Start — D0
Ext Length in BYTEs — D0
Ext Max Length in BYTEs — 4

Device – Designates which of the pre-configured EtherNet/IP Client devices to use when sending the message. Part of the configuration for a device is assigning a name to the device. It is that name which will show up in the Device selection drop-down menu. For more information on configuring devices go to the Help file.

IP Address – The IP Address of the EtherNet/IP Server (Slave) to send the message to. This can be either a Fixed (static) IP Address or a Variable (dynamic) value as described below:

Fixed Address – The TCP/IP Address assigned to the EtherNet/IP Server (Slave). IP addresses are canonically represented in dot-decimal notation, consisting of four decimal numbers, each ranging from 0 to 255, separated by dots.

NOTE: *Invoking the Element Browser (F9) for this field will bring up the IP Address Lookup utility that can find the IP Address for a given name.*

Variable IP Address – The IP Address resides in the specified memory location. This can be any readable DWord numeric location.

TCP Port Number – The port number of the EtherNet/IP Server (Slave) to send the message to. The default value of 44818 is typically the correct port number for EtherNet/IP protocol. This can be any constant value between 0 and 65535, or any readable numeric location containing a value in that range.

EtherNet/IP Client (Master), continued

Path – Specifies the parameters for the request. The specific values needed for the fields will be provided by the manufacturer of the EtherNet/IP server that you are talking to.

Class – The Class ID value (defined by the EtherNet/IP Server). This can be any positive integer value or any readable numeric location.

Instance – The Instance ID value (defined by the EtherNet/IP Server). This can be any positive integer value or any readable numeric location.

Use Attribute – Enable this option to specify the Attribute value (defined by the EtherNet/IP Server). This can be any positive integer value or any readable numeric location.

Service - Specifies the operation to perform on the set of objects. Choose from the following list of predefined Services, or select Generic and enter the Service number.

Specific Service – Select one of the predefined Service Requests below:

Get Single Attribute (14, 0x0E) - request a single attribute

Set Single Attribute (16, 0x10) - write a single attribute

Get All Attributes (1, 0x01) - request all of the attributes

Set All Attributes (2, 0x02) - write to all of the attributes

Generic Service – Specify a Service that is NOT one of the predefined Service Requests. This can be any constant integer value or readable memory location.

Create Data Block – If an appropriate data block does not already exist, or if you want to create an additional data block for use in this instruction, then click this button to open a dialog where you can create a new data block of the required type.

Use Request Service Data Buffer – This selection will be automatically enabled when any Set Attribute service is selected and automatically disabled when any Get Attribute service is selected. This buffer can be enabled any time the Generic Service is selected.

Req is String Structure – Select this option if the data for the Set Attribute service or any Generic service is contained in a String, then enter the String element to use. This can be any of the system-defined Short Strings, or system-defined Long Strings, or any of the user-defined Strings. The maximum length of the String to send is 500 bytes.

Req is Numeric Data Block – Select this option if the data for the Set Attribute service or any Generic service is contained in a numeric memory block. The maximum size of the data block that can be sent in a single service request 500 bytes (250 Words, 125 DWords, 125 Reals).

Req Start – Specify the first element of the memory block that is the data for the Set Attribute or Generic service.

Req Number of Bytes - The number of consecutive BYTEs of data for the Set Attribute or Generic service (Words = 2 Bytes, DWord = 4 Bytes, Real = 4 Bytes).

EtherNet/IP Client (Master), continued

Use Response Service Data Buffer – This selection will be automatically enabled when any Get Attribute service or any Generic service is selected and automatically disabled when any Set Attribute service is selected. This buffer can be enabled any time the Generic Service is selected.

Res is String Structure – Select this option to store the data from the Get Attribute service or any Generic service in a String, then enter the String element to use. This can be any of the system-defined Short Strings, or system-defined Long Strings, or any of the user-defined Strings. The maximum length of the String that could potentially be received is 500 bytes, so make sure the String can handle the maximum response for the desired service.

Res is Numeric Data Block – Select this option to store the data from the Get service or any Generic service in a numeric memory block. The maximum size of the data block that can be read in a single service request 500 bytes (250 Words, 125 DWords, 125 Reals).

Res Start – Specify the first element in the numeric data block to store the data that was returned by the Get Attribute or Generic service. This can be any writable numeric location.

Res Length in BYTEs – Specify a memory location to store the actual number of Bytes of data that was returned by the Get Attribute or Generic service. This can be any writable numeric location.

Res Max Length in BYTEs – Specify the maximum number of BYTEs of the returned data to retain to store in the data block. This can be any positive integer constant between 1 and 500 or any readable numeric location.

***NOTE:** The byte length value should be a multiple of the number of BYTEs in a single element in the numeric memory block. For example, if the memory block consists of DWords or Reals, this value should be a multiple of 4, as there are 4 BYTEs per DWord or Real.*

General Status Code – enable this option to store the value returned from the EtherNet/IP Server in response to processing the Service Request; enter the numeric location to store the value. This can be any writable numeric location. This value could indicate success or be an error code. Consult the documentation for the EtherNet/IP Server for information on how to interpret General Status Code values.

Extended Status – Enable this option to store any extended status value returned from the EtherNet/IP Server in response to processing the Service Request.

Ext is String Structure – Select this option to store the extended status information in a String, then enter the destination String element. This can be any of the system-defined Short Strings, or system-defined Long Strings, or any of the user-defined Strings.

Ext is Numeric Data Block – Select this option to store the Extended Status data in a numeric data block.

Ext Start – Specify the first element in the numeric data block to store the Extended Status data. This can be any writable numeric location.

Ext Length in BYTEs – Specify a memory location to store the actual number of Bytes of Extended Status data. This can be any writable numeric location.

Ext Max Length in BYTEs – Specify the maximum number of BYTEs of Extended Status data to store in the data block. This can be any positive integer constant between 1 and 500, or any readable numeric location.

EtherNet/IP Client (Master), continued

***NOTE:** The byte length value should be a multiple of the number of BYTEs in a single element in the numeric memory block. For example, if the memory block consists of DWords or Reals, this value should be a multiple of 4, as there are 4 BYTEs per DWord or Real.*

Enable – Designates how this instruction will operate. Select from one of the following:

Once on Leading Edge - Select this option to have this instruction run to completion exactly one time. Typically, this instruction will take more than one controller scan to complete. Configured this way the instruction is Edge Triggered.

Continuous on Power Flow at Interval - Select this option to have this instruction run as long as the instruction has power flow. After the instruction has initially run, if the instruction still has power flow, the instruction will remain enabled and will wait the specified amount of time before running again. The following options select how much time (in milliseconds) to wait between successive runs. A value of zero milliseconds (0ms) means the instruction will re-run immediately.

On Success – When the instruction completes successfully this action will be performed.

Set Bit – Enable this selection then specify writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

On Error Counter – Enable this option and select a DWord location to store the total number of times the RX failed to complete. This can be any DWord location.

On Error – When the instruction does not complete successfully this action will be performed.

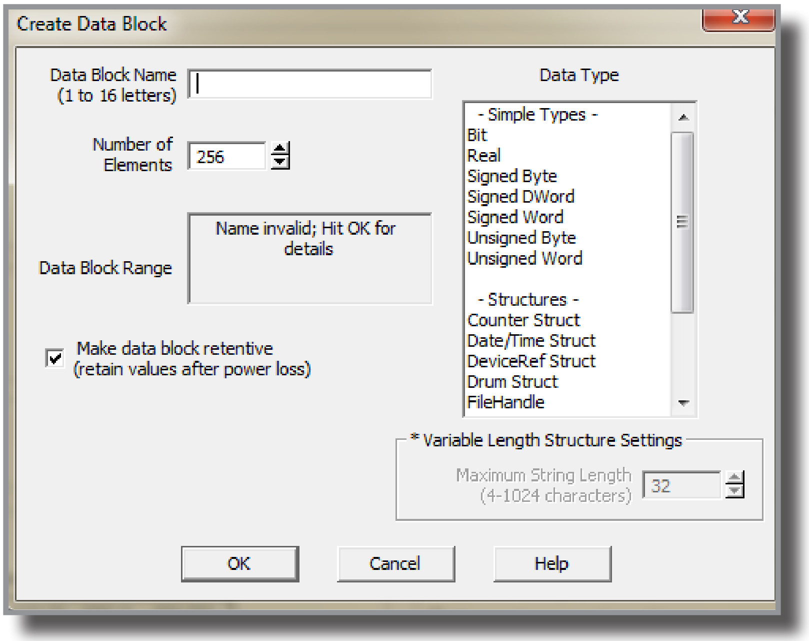
Set Bit – Enable this selection then specify writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

On Error Counter – Enable this option and select a DWord location to store the total number of times the RX failed to complete. This can be any DWord location.

EtherNet/IP Client (Master), continued

Create Data Block – If an appropriate data block does not already exist, or if you want to create an additional data block for use in this instruction, then click this button to open a dialog where you can create a new data block of the required type.


Create Data Block

Data Block Name (1 to 16 letters)

Number of Elements 256

Data Block Range: Name invalid; Hit OK for details

[✓] Make data block retentive (retain values after power loss)

Data Type
- Simple Types -
Bit
Real
Signed Byte
Signed DWord
Signed Word
Unsigned Byte
Unsigned Word
- Structures -
Counter Struct
Date/Time Struct
DeviceRef Struct
Drum Struct
FileHandle

* Variable Length Structure Settings
Maximum String Length (4-1024 characters) 32

Data Block Name (1 to 16 letters) – Block names must be unique, and consist of 1 to 16 characters (A-Z, a-z; no numbers, no spaces).

Number of Elements – Specifies the number of bytes in the data block. The data blocks must be created on a DWord (4-byte) boundary. The maximum number of Bytes that can be received from a single packet is 1024.

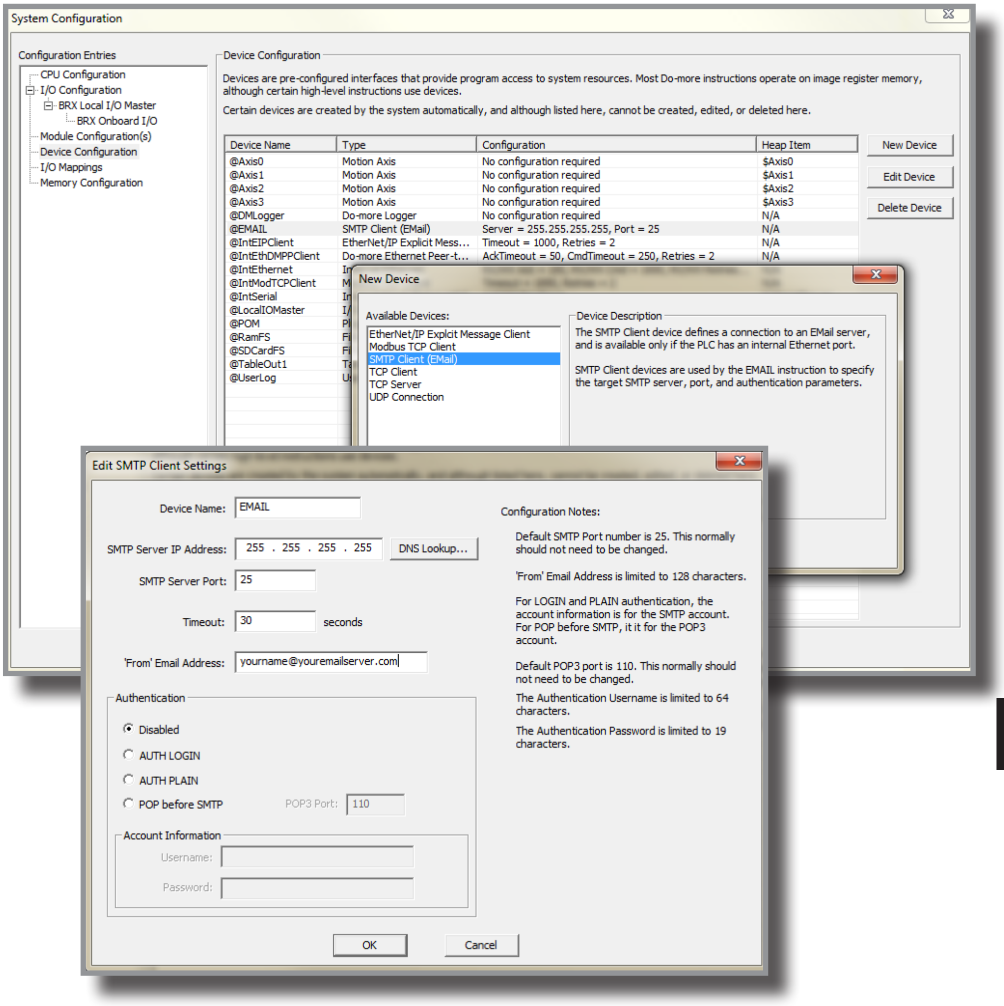
Data Block Range – Displays the first and last element of the block that will be created based on the current entries for Data Block Name and Number of Elements.

Data Type – The data block will consist of Unsigned Bytes.

Make Data Block Retentive (retain values after power loss) – A data block marked as retentive will hold its state through a power cycle or a Program-to-Run mode transition. The status of memory NOT marked as retentive will be cleared at power up and during a Program-to-Run mode transition.

SMTP – EMAIL

The Edit SMTP Client Settings dialog is used to configure an SMTP connection from the Do-more! controller to an SMTP server in order for the controller to send Email. The information that is required to configure an SMTP connection is always going to be specific to the installation. It is up to the programmer to locate the required information.



Device Configuration

Devices are pre-configured interfaces that provide program access to system resources. Most Do-more instructions operate on image register memory, although certain high-level instructions use devices.

Certain devices are created by the system automatically, and although listed here, cannot be created, edited, or deleted here.

Device Name	Type	Configuration	Heap Item
@Axis0	Motion Axis	No configuration required	$Axis0
@Axis1	Motion Axis	No configuration required	$Axis1
@Axis2	Motion Axis	No configuration required	$Axis2
@Axis3	Motion Axis	No configuration required	$Axis3
@DMLogger	Do-more Logger	No configuration required	N/A
@EMAIL	SMTP Client (EMail)	Server = 255.255.255.255, Port = 25	N/A
@IntEIPClient	EtherNet/IP Explicit Mess...	Timeout = 1000, Retries = 2	N/A
@IntEthDMPPClient	Do-more Ethernet Peer-t...	AckTimeout = 50, CmdTimeout = 250, Retries = 2	N/A
@IntEthernet	In		
@IntModTCPClient	M		
@IntSerial	In		
@LocalIOMaster	I/		
@POM	Pl		
@RamFS	Fi		
@SDCardFS	Fi		
@TableOut1	Ta		
@UserLog	U		

New Device

Available Devices:

- EtherNet/IP Explcit Message Client
- Modbus TCP Client
- SMTP Client (EMail)
- TCP Client
- TCP Server
- UDP Connection

Device Description

The SMTP Client device defines a connection to an EMail server, and is available only if the PLC has an internal Ethernet port.

SMTP Client devices are used by the EMAIL instruction to specify the target SMTP server, port, and authentication parameters.

Edit SMTP Client Settings

Device Name:	EMAIL
SMTP Server IP Address:	255 . 255 . 255 . 255
SMTP Server Port:	25
Timeout:	30 seconds
'From' Email Address:	yourname@youremailserver.com

Authentication

- (●) Disabled
- ( ) AUTH LOGIN
- ( ) AUTH PLAIN
- ( ) POP before SMTP   POP3 Port: 110

Account Information

Username:

Password:

Configuration Notes:

Default SMTP Port number is 25. This normally should not need to be changed.

'From' Email Address is limited to 128 characters.

For LOGIN and PLAIN authentication, the account information is for the SMTP account. For POP before SMTP, it it for the POP3 account.

Default POP3 port is 110. This normally should not need to be changed.

The Authentication Username is limited to 64 characters.

The Authentication Password is limited to 19 characters.

SMTP – EMAIL, continued

Device Name – The name given to the SMTP Client. This is the name that will be referenced in the Send Email (EMAIL) instructions. Device Names can consists of 1 to 16 alphanumeric characters. Device names must follow Nickname rules.

SMTP Server IP Address – The IP address of the SMTP server that will process the Email sent from the controller. If the IP Address of your SMTP Server is not static, you can use the DNSLOOKUP and DEVWRITE instructions to determine it at runtime before executing an EMAIL instruction. See the example in the EMAIL Help Topic DMD0068. If this is the case, just fill in a dummy IP Address in the SMTP Configuration.

Clicking the DNS Loopkup button will open the IP Address Lookup utility so that Do-more! Designer can search for the IP Address assigned to a given SMTP Server name. This requires a functional connection to a DNS Server.

NOTE: *The IP Address of the SMTP Server can be resolved at runtime by using the Name to IP Address (DNSLOOKUP) instruction to find the IP Address associated with a Server's name, then use the Write Device Register (DEVWRITE) instruction to set the SMTP Client Device to use the IP Address that was found.*

SMTP Server Port – The default value of 25 is the standard IP port number that is used by SMTP servers. The SMTP Server Port number can be examined at runtime with the Read Device Register (DEVREAD) instruction, and changed at runtime through the Write Device Register (DEVWRITE) instruction.

Timeout –This is the amount of time (in seconds) the SMTP Client will attempt to connect to the specified SMTP server before reporting an error. The default value of 30 seconds should be sufficient in most instances. Be aware that it is quite normal for communications with SMTP servers to take several seconds of time; setting this value too low will only cause needless problems.

From Email Address – Specifies the Email address that all Emails using this SMTP Client will use in the 'From' field. Email addresses must be in the form of X@Y.Z. SMTP servers typically require that the 'From' address be configured as a recipient address on that SMTP server before they will accept Emails from that address.

Authentication – If the SMTP server requires authentication before it will accept an Email from the controller, select one of the following three methods:

Disabled – If the SMTP server does not require authentication

AUTH LOGIN – Authenticate by logging into the SMTP Server with the Username and Password specified below

AUTH PLAIN – Authenticate by logging into the SMTP Server with the Username and Password specified below

POP before SMTP – An authentication that attempts to get Email before attempting to send an Email, the premise being that if an Email client can log in and read Email the Client must be legitimate

Pop3 Port – The default port number of 110 is the standard IP port number for POP3 requests and should not need to be change. The Pop3 Port number can be examined at runtime with the Read Device Register (DEVREAD) instruction, and changed at runtime through the Write Device Register (DEVWRITE) instruction.

SMTP – EMAIL, continued

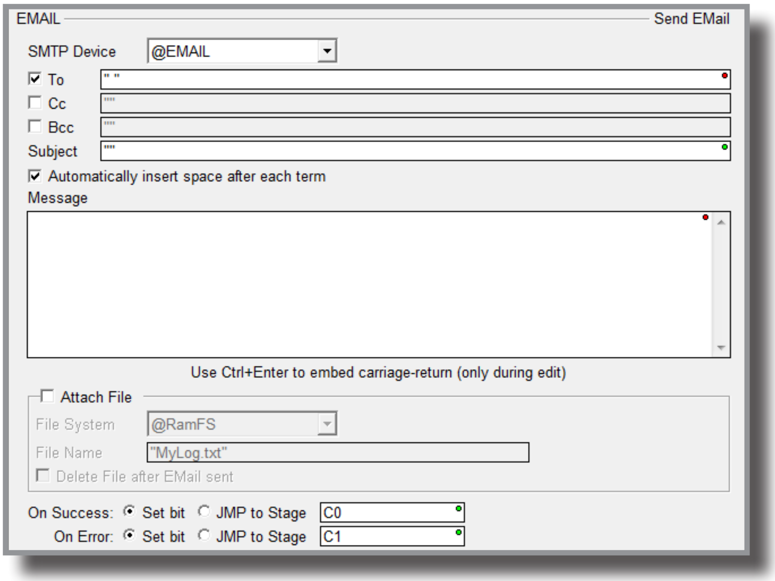
Account Information – The three authentication methods above require a UserID and Password. For 'AUTH LOGIN' and 'AUTH PLAIN', the account information is for the SMTP account, for 'POP before SMTP' method, the account information is for the POP3 account.

Username - 1 to 64 characters

Password - 1 to 19 characters

EMAIL

The Send Email (EMAIL) instruction is used to send an Email message. This instruction is only valid for Do-more! controllers that have an on-board Ethernet port. The message portion of the Email can be any combination of text and data elements from the controller. An Email can also send an attachment which can be any file on any of the built-in file systems.


EMAIL — Send EMail

SMTP Device: @EMAIL

[✓] To: " "

[ ] Cc: ""

[ ] Bcc: ""

Subject: ""

[✓] Automatically insert space after each term

Message:

Use Ctrl+Enter to embed carriage-return (only during edit)

[ ] Attach File

File System: @RamFS

File Name: "MyLog.txt"

[ ] Delete File after EMail sent

On Success: [✓ Set bit] JMP to Stage — C0

On Error: [✓ Set bit] JMP to Stage — C1

SMTP Device – Designates the SMTP Client device to use to send the Email. An SMTP Client device must be configured before the Send Email instruction can be added to a program.

To – One or more primary audience Email addresses separated by semicolons. This can be a string literal (text in double quotes), or any String element.

Cc (optional) – One or more Courtesy Copy (or Carbon Copy) Email addresses separated by semicolons. This can be a string literal (text in double quotes), or any String element.

Bcc (optional) – One or more Blind Carbon Copy Email addresses separated by semicolons. This can be a string literal (text in double quotes), or any String element.

Subject – The Total number of characters in Subject line is limited to 1024 characters. This can be a string literal (text in double quotes), or any readable String element.

Automatically insert space after each term – Will insert a space between the terms when the instruction is processed. This is most useful when the Message Field contains only a list of elements that would otherwise require a manually entered space character to separate the items.

EMAIL, continued

Message – Text box in which to place the body of the Email message. There can be up to 1023 characters of data in a Message field. This data can consist of any combination of the following:

String Literal (text in double quotes).

Control characters.

Any readable String element.

Controller data elements (V0, D0, T0.Acc, etc.).

Data formatting functions (FmtInt, FmtReal, etc.).

String selection function (Lookup).

For a complete description of the available data options provided by the scripting language for use in the Text Field, see the Help file.

Attach File – Enable this option to send an existing file from one of the file Systems in the Do-more CPU along with the Message text.

File System – Specifies which of the available file systems contains the file.

@RamFS – The 1 MB file system in the Do-more CPU's system RAM. All Do-more CPUs will have this file system available.

@SDCardFS – On PLC systems that have the microSD card slot, this selection is the file system on the removable media in that slot.

File Name – The full path (including any directories) of the file on the specified file system. This can be text enclosed in double quotes, or any system or user-defined string. The File Name allows a maximum length of 255 characters including spaces and non-alphanumeric characters, excluding the following characters which have special meaning to the file system * **?** " **:** < >. The File Name is not case sensitive.

Delete File After Email Sent – Enable this option to delete the specified file AFTER the Email has been successfully sent to the specified SMTP Server.

On Success – When the instruction completes successfully this action will be performed.

Set Bit – Enable this selection then specify writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

On Error – When the instruction does not complete successfully this action will be performed

Set Bit – Enable this selection then specify writable bit location.

JMP to Stage – Enable this selection then specify any Stage number from S0 to S127 in the current Program code-block.

Notes: